Claude Delannoy

dBASE II

SUR

AMSTRAD

EYROLLES

dBASE II
SUR
AMSTRAD

CHEZ LE MEME EDITEUR

Du même auteur :	EXERCICES DE BASIC
-	JE DEBUTE EN BASIC AMSTRAD
-	FAITES VOS JEUX AVEC AMSTRAD
-	MULTIPLAN SUR AMSTRAD
-	LES FICHIERS EN BASIC SUR MICRO ORDINATEUR
-	INITIATION A MULTIPLAN Version 1 et 2, avec exercices et corrigés
P. Beaufils, M. Lamarche et Y. Muggianu	PROGRAMMES DE PHYSIQUE SUR AMSTRAD
-	PROGRAMMES DE MATHEMATIQUES SUR AMSTRAD
P. Bihan	PROGRAMMATION SUR AMSTRAD PCW 8256/8512 Basic et fichiers
-	FICHIERS SUR AMSTRAD PCW 8256/8512 Basic JETSAM et CP/M
M. Rousselet	CALCUL NUMERIQUE SUR AMSTRAD

dBASE II
SUR
AMSTRAD

Claude Delannoy

EYROLLES

61, boulevard Saint-Germain – 75005 PARIS
1987

Si vous désirez être tenu au courant de nos publications, il vous suffit d'adresser votre carte de visite au :

Service « Presse », Editions EYROLLES,
61, Boulevard Saint-Germain
75240 PARIS CEDEX 05,

en précisant les domaines qui vous intéressent. Vous recevrez régulièrement un avis de parution des nouveautés en vente chez votre libraire habituel.

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40). »

« Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. ».

© Editions EYROLLES, 1987

Avant-Propos

La puissance et la souplesse de dBASEII(*) en font l'un des plus célèbres logiciels de ''gestion de bases de données''. L'objectif de ce livre est de vous amener à maîtriser progressivement cet ''outil'' et de vous permettre ainsi de l'utiliser avec profit pour vos applications personnelles.

De par sa nature même, un tel logiciel s'adresse à des personnes diverses, tant par leur formation ou leur connaissances que par leur domaine d'activité. C'est pourquoi nous avons volontairement écarté tout jargon spécifique d'une profession et nous ne supposons aucune connaissance préalable, fût-ce en informatique. Dans le même esprit, les fichiers servant de support à nos exemples ont été choisis pour leur simplicité et leur caractère universel.

Guidés par notre expérience de l'enseignement, nous avons choisi ici une démarche du type ''apprentissage''. Tout d'abord, chaque nouvelle notion est systématiquement accompagnée d'exemples simples ; présentés sous forme ''d'écrans'', ils peuvent être immédiatement essayés sur votre Amstrad.

Ensuite et surtout, nous vous proposons de très nombreuses manipulations autour d'un thème. Celles-ci sont repérées par le titre

(*) dBASE II est une marque déposée de Ashton Tate.

''Entraînez-vous'' et le filet qui les borde en marge. Leur but et de vous familiariser avec la conduite de dBASEII en vous faisant expérimenter des situations élémentaires très variées. Les possibilités d'erreur y sont minutieusement explorées : leur détection et leur identification, lors de la réalisation de vos propres applications, en sera d'autant plus aisée que vous aurez fait leur connaissance dans un contexte simple.

Nous ne saurions trop vous conseiller de ne pas négliger ces ''travaux pratiques''. C'est grâce à eux que vous acquièrerez le ''savoir faire'' indispensable à une utilisation efficace et rationnelle de dBASE II.

Ce livre peut être considéré comme formé de deux parties :

Les chapitres 1 à 13 présentent l'essentiel des commandes de dBASE II utilisables en ''mode conversationnel''. Nous y avons volontairement omis certains aspects secondaires du logiciel, au profit d'une présentation fonctionnelle et pédagogique (Ceux-ci sont exposés dans le chapitre 18).

Les chapitres 14 à 17 constituent une initiation à la programmation en dBASE II. Vous y découvrirez comment le ''langage structuré'' que vous offre ce logiciel vous permet d'en démultiplier la puissance. Vous serez ainsi en mesure, non seulement d'automatiser ou de personnaliser vos propres applications, mais également et surtout d'aborder la réalisation d'applications ''clés en main'' livrables à n'importe quel utilisateur néophyte.

Table des matières

Avant-propos VII

I. Des fichiers manuels aux bases de données : le rôle de dBASE II 1

1. Qu'est-ce qu'un fichier manuel ? 2
2. Quelles opérations réalise-t-on sur un fichier manuel ? 2
3. Qu'est-ce qu'un fichier pour dBASE II ? 3
4. Quelles opérations réalise-t-on avec un fichier sous dBASE ? . 4

II. Contact 6

1. Pour faire des copies de sauvegarde 6
2. Pour aborder l'étude de ce manuel 7
3. Pour démarrer dBASE II 8

III. Créons notre premier fichier 11

1. Comment créer un fichier en dBASE 12
2. Pour définir la structure d'un fichier 12
3. Pour connaître les fichiers existants : LIST FILES 14
4. Quelques règles à respecter 15
5. Quand dBASE ne reconnaît pas votre commande 16
6. Pour faire du ''ménage'' : DELETE FILE 17
7. Comment remplir un fichier 18

IV. Pour voir ce que nous avons créé 21

1. Pour connaître le contenu d'un fichier : LIST 21
2. Pour retrouver la structure d'un fichier : LIST STRUCTURE ... 23

3. Notion de "zone de travail" 24
4. Toutes les commandes ne portent pas sur la "zone de travail" 24
5. Quand aucun lien n'a été établi 26
6. Un seul lien à la fois 27
7. Pour obtenir des informations permanentes : utiliser l'imprimante 29

V. La mise à jour "à vue" 30

1. Pour modifier le contenu de certains enregistrements : EDIT .. 31
2. Pour ajouter des enregistrements en fin de fichier : APPEND . 33
3. Pour voir un enregistrement particulier 34
4. Pour supprimer un enregistrement 35
5. En cas de remords, pour supprimer des marques d'effacement : RECALL 38
6. Plusieurs façons de marquer un enregistrement pour effacement 39
7. Pour insérer un enregistrement 40

VI. Consulter un fichier 41

1. Consultation globale ou restreinte 41
2. Pour n'afficher que certains champs 44
3. Pour afficher des "expressions" 45
4. Pour expérimenter vos propres expressions 51

VII. Pour imposer vos conditions 53

1. Pour mettre des conditions : FOR 53
2. Les conditions les plus usitées : les comparaisons 55
3. L'exactitude des comparaisons d'égalité pour les caractères : SET EXACT 57
4. La condition d'appartenance : $ 59
5. Pour retrouver un par un les enregistrements vérifiant une condition : LOCATE FOR et CONTINUE 59
6. Les conditions multiples 60
7. Pour compter le nombre d'enregistrements vérifiant une condition : COUNT 62
8. Pour faire des sommes : SUM 63

VIII. La mise à jour "automatique" 65

1. Pour recopier des fichiers : COPY TO 65
2. Pour effacer plusieurs enregistrements avec DELETE 67
3. La commande DELETE avec indication de "portée" 67
4. Effacement conditionnel 68
5. Pour effectuer des remplacements automatiques : REPLACE . 69
6. La commande REPLACE avec indication de "portée" 71
7. Remplacements conditionnels 73

IX. Le tri : une (certaine) façon de mettre de l'ordre 75

1. Pour trier sur un champ de type caractère 76
2. Quel ordre pour les caractères ? 77
3. Quand des chiffres se trouvent dans un champ caractère 78
4. Pour trier sur un champ de type numérique 79

X. L'indexation : une (autre) façon de mettre de l'ordre et de gagner du temps 82

1. Notion d'index en dBASE 82
2. Comment créer un index : INDEX ON... TO... 84
3. Pour utiliser un index existant 85
4. Le pointeur se déplace suivant l'index 86
5. Pour retrouver un enregistrement de clé donnée : FIND 88
6. FIND ou LOCATE ? 91
7. Quand l'index est automatiquement mis à jour 92

XI. Pour en savoir plus sur l'indexation 95

1. Pour que dBASE réorganise automatiquement plusieurs index 95
2. Pour mettre à jour des index 97
3. Quand la clé est une expression 97
4. Pour savoir où vous en êtes 100

XII. Edition de rapports 104

1. Création d'un rapport simplifié — la commande REPORT 104
2. Le fichier "FORMAT" 106
3. Rapport avec sélection 108
4. Pour faire des calculs dans les rapports : expressions et totaux 109
5. Pour obtenir des "totaux partiels" 111

XIII. Pour modifier la structure d'un fichier 115

1. Un cas particulier de recopie : COPY STRUCTURE TO... 116
2. Pour ajouter "automatiquement" des enregistrements à un fichier : APPEND FROM 117
3. Pour modifier la structure d'un fichier : MODIFY STRUCTURE 120
4. Pour ajouter un champ à un fichier existant 122
5. Pour supprimer un champ d'un fichier 124
6. Pour modifier la taille d'un champ 124
7. Pour modifier le nom d'un champ 125

XIV. Premières notions de programmation 126

1. Créer et exécuter un programme 127
2. Pour "revoir" un programme 129
3. En cas d'erreur de programme 129
4. La notion de variable 130
5. Le type d'une variable 133
6. Pour connaître toutes les variables utilisées : DISPLAY MEMORY 134
7. Pour supprimer des variables inutiles : RELEASE 135
8. Pour conserver dans une variable les résultats de COUNT et SUM 136
9. Un programme utilisant des variables 137
10. Pour communiquer des informations à un programme : ACCEPT et INPUT 138

XV. Pour qu'un programme prenne des décisions : les "structures" de choix : IF... et DO CASE... 143

1. La "structure" de choix : IF... ELSE... ENDIF 143
2. Un programme à double usage 146

3. Un programme de recherche 147
4. Quelques règles à propos de la structure de choix 149
5. La structure de choix multiple : DO CASE... ENDCASE 150
6. Pour les "imprévus" : OTHERWISE 152
7. Un programme "à menu" 153
8. Pour y voir plus clair dans vos programmes : les commentaires 156

XVI. Pour répéter des commandes : la structure de boucle 158
1. La structure de boucle : DO WHILE... ENDDO 158
2. Pour répéter un traitement sur tous les enregistrements d'un fichier . 161
3. Pour vous aider à détecter vos erreurs 163

XVII. Pour bien gérer l'écran 167

1. Pour afficher à votre guise : à SAY 167
2. Un programme de liste "plein écran" 168
3. Pour programmer vos saisies en mode "plein écran" : GET et READ 170
4. Pour contrôler vos saisies 172

XVIII. A vous de jouer 176

1. La commande BROWSE 177
2. La commande CHANGE... FIELDS 177
3. Les Macros 177
4. Conservation des valeurs des variables 177
5. USING et PICTURE 178
6. Les fonctions 178
7. Les indicateurs qu'il est possible d'activer par SET 178
8. Zone primaire et zone secondaire 178
9. La commande JOIN 179

Correction des "entraînez-vous" 180

Index 184

I Des fichiers manuels aux bases de données : le rôle de dBASE II

Vous savez probablement ce qu'est un ''fichier'' au sens usuel (et manuel) du terme ! Il vous suffit pour cela de penser à un répertoire d'adresses, à un fichier du personnel, à des fiches de prêt de bibliothèque, à un livre comptable, à un fichier clients, stock,...

Vous avez également, sans doute, certaines idées et aspirations concernant ce que vous pouvez attendre d'un logiciel comme dBASE II associé à votre Amstrad. Vous devinez qu'il va vous permettre de faciliter certaines de vos tâches de gestion de fichiers (ou bases de données) en les rendant plus rapides et plus fiables et en vous offrant des possibilités nouvelles. Peut-être même brûlez-vous d'impatience de passer à l'action !

C'est effectivement ce que nous allons faire dès le prochain chapitre. Auparavant, nous souhaitons vous donner quelques idées générales sur ce qu'est un fichier en dBASE II et vous montrer que sa différence avec un fichier manuel ne se limite pas au fait qu'il est ''traité par ordinateur''.

1. QU'EST-CE QU'UN FICHIER MANUEL ?

Il est toujours difficile de donner une définition précise de choses qui nous sont familières. Néanmoins, nous pouvons dire qu'un fichier manuel est une collection ''structurée'' d'informations. Pourquoi structurée ? Parce que n'importe quel ensemble d'informations ne peut être qualifié de fichier. Il ne viendrait probablement à l'esprit de personne de dire qu'un journal constitue un fichier !

A quoi correspond en pratique cette ''structuration'' de l'information d'un fichier ? Très souvent au découpage en un ensemble de ''*fiches*''. Mais, direz-vous, un répertoire téléphonique n'est pas un ensemble de fiches ! Celà est vrai, sur un plan matériel ; Néanmoins, il serait facile d'imaginer que toutes les informations relatives à une personne ont été portées sur une fiche !

La fiche constitue donc une première façon de structurer un fichier. Il en existe une seconde, au niveau de la structure de la fiche elle-même. Généralement, toutes les fiches comportent les mêmes *rubriques* ; par exemple, dans le cas d'un répertoire, ces rubriques seraient : nom, prénom, adresse, téléphone.

En résumé, donc, nous pouvons dire qu'un fichier manuel est formé d'un ensemble de fiches, formées elles-mêmes de rubriques.

2. QUELLES OPÉRATIONS RÉALISE-T-ON SUR UN FICHIER MANUEL ?

En général, nous commençons par définir un ''modèle de fiche'' précisant les rubriques que nous souhaitons y voir figurer. Cette opération peut avoir un caractère purement abstrait : nous décidons que nous ferons figurer telle ou telle rubrique, mais cela n'entraîne aucune action effective, au niveau du fichier. Au contraire, cette opération peut entraîner la préparation de tout un lot de fiches, sur lesquelles nous portons par exemple des ''titres de rubriques'', ou encore sur lesquelles nous préparons des emplacements. Cette opération peut également revêtir un caractère purement implicite : par exemple, dans le cas d'un répertoire, les rubriques sont déjà prévues sur le ''support'' lui-même ; il n'y a donc rien à faire et le choix des rubriques nous est imposé.

Nous sommes ensuite amenés à remplir tout ou partie des fiches. A ce stade, nous pouvons dire que nous avons **créé** un fichier.

Que fait-on avec un fichier, une fois qu'il a été créé ? Nous pouvons :

a) Le consulter pour retrouver l'information portée sur une fiche. La façon de procéder pour retrouver la ''bonne fiche'' est généralement laissée à la discrétion de l'utilisateur qui peut, par exemple :

— chercher au hasard
— explorer systématiquement les fiches les unes après les autres
— tenir compte d'un certain ordre de rangement des fiches pour accélérer sa recherche (dans certains cas, le fichier pourra alors être ''indexé'')
— ...

b) En extraire de l'information, en explorant tout le fichier. Par exemple, à partir d'un fichier répertoire, on pourra :

— établir la liste de toutes les personnes habitant PARIS
— compter le nombre de personnes prénommées Albert
— ...

c) Le ''mettre à jour'', c'est-à-dire effectuer des modifications qui peuvent être de trois types :

— Modifier le contenu d'une fiche, soit pour corriger une erreur, soit pour tenir compte d'une évolution de l'information (par exemple : changement d'adresse)
— Ajouter de nouvelles fiches
— Supprimer des fiches devenues inutiles

3. QU'EST-CE QU'UN FICHIER POUR dBASE II ?

Tout d'abord, nous allons retrouver sous dBASE II certaines analogies avec les fichiers manuels, avec néanmoins une transformation du vocabulaire, comme l'indique ce tableau :

Terme utilisé pour des fichiers ''manuels''	Terme utilisé avec dBASE II
Fichier	Fichier(*) Base de données
Fiche	Enregistrement
Rubrique	Champ Rubrique
Modèle de fiche	Structure

(*) En théorie, les termes Ficher et Base de données ont des significations différentes.

Pour Dbase (*), un fichier reste donc un ensemble d'enregistrements, comportant les mêmes champs (ou rubriques). Par contre, cette fois, il n'y aura plus de place pour l'approximatif : si, dans un fichier manuel, on pouvait éventuellement imaginer ajouter dans des fiches des informations complémentaires pour lesquelles aucune rubrique n'avait été prévue, il n'en sera plus question avec Dbase. Ainsi, un fichier peut être considéré comme un tableau dont les lignes seraient les enregistrements et les colonnes les champs.

Voici à quoi pouvait ressembler, sous Dbase un fichier répertoire simplifié comportant simplement deux champs (nom et téléphone) et quatre enregistrements

nom	téléphone
Thomas	86 48 23 37
Mitenne	89 55 66 89
Loiseau	84 61 28 42
Meunier	77 89 63 10

4. QUELLES OPÉRATIONS RÉALISE-T-ON AVEC UN FICHIER SOUS dBASE II ?

La définition d'un modèle de fiche telle qu'elle était prévue pour un fichier manuel deviendra la **définition de la structure.** Mais, cette fois, nous devrons, non seulement définir chacun des champs (nous verrons que nous aurons à leur choisir un nom) mais en outre, leur attribuer une **taille.** Autrement dit, dans l'exemple précédent d'un fichier répertoire simplifié, il nous faudra définir la largeur des colonnes. Celle-ci imposera nécessairement des limites à l'information que nous pourrons ultérieurement y placer. (Notez qu'avec un fichier manuel, ce type de limitations était beaucoup moins rigide...).

Le remplissage des fiches portera souvent le nom de **saisie.**

La **consultation** pourra se faire de différentes manières :

a) à partir du numéro d'enregistrement. En effet, les enregistrements porteront un numéro correspondant à l'ordre de leur création. Dbase vous permettra de retrouver *instantanément* un enregistrement de numéro donné (on appele cela : accès direct). Mais n'oubliez pas tou-

(*) Il nous arrivera souvent d'abréger dBASE III en Dbase.

tefois que pour bénéficier de cet accès rapide, il faudra être en mesure de fournir le numéro de l'enregistrement en question ! (Pensez, par exemple, à un fichier répertoire et demandez-vous s'il est très pratique de parler de son 259e enregistrement !).

b) à partir du contenu d'un enregistrement : Dbase vous permettra de retrouver un enregistrement à partir de la connaissance du contenu de l'un de ses champs. Par exemple, vous pourrez, à partir de notre fichier répertoire précédent, obtenir le nom de la personne ayant un numéro de téléphone donné. Certes, la consultation ne se fera plus ici aussi rapidement qu'à partir du numéro d'enregistrement (il faut bien que Dbase explore le fichier). Toutefois, nous verrons comment l'indexation permettra de rendre quasi instantanée ce genre de recherche.

En ce qui concerne **l'extraction d'information,** Dbase va se révéler très puissant. Il vous permettra de réaliser facilement, rapidement et sans erreur des opérations telles que celles que nous avions évoquées avec un fichier manuel (liste des habitants de PARIS, nombre de personnes prénommées Albert). Vous pourrez y introduire des conditions de sélection presque aussi complexes que vous le souhaiterez (par exemple tous les Albert, habitant LYON dont le numéro de téléphone comporte le groupement 86).

La **mise à jour** comportera bien sûr les trois possibilités déjà évoquées (modification d'enregistrements, ajout ou suppression d'enregistrements). Mais Dbase vous permettra de les faire porter automatiquement sur tout un ensemble d'enregistrements. Par exemple, dans un fichier comportant des prix de ventes, vous pourrez demander à Dbase d'augmenter tous les prix de 10 % (imaginez ce que cela signifierait pour un fichier manuel de 2 000 fiches !).

5. LES DEUX FAÇONS D'UTILISER dBASE

Pour manipuler vos fichiers, Dbase vous offre d'abord un vaste éventail de ''**commandes**''. Il s'agit d'ordres que vous lui fournissez au clavier et qu'il exécute sur le champ. (Les chapitres 2 à 13 sont consacrés à l'apprentissage des principales commandes).

Mais, en outre, Dbase vous offre un véritable **langage de programmation.** Les programmes que vous pourrez ainsi réaliser deviendront en quelque sorte des ''super-commandes'' que vous vous serez taillées sur mesure. En outre, c'est notamment grâce à la programmation que Dbase révélera ses possibilités de gestion simultanée de plusieurs fichiers, entre lesquels un ''champ commun'' permet d'établir des ''relations''.

II Contact !

Le logiciel dBASE II vous est fourni sur une disquette convenant aussi bien au CPC 6128 qu'au CPW 8256 ou au CPW 8512. L'une des faces contient le logiciel proprement dit, l'autre face est consacrée à des ''exemples''.

Dans ce (petit) chapitre, nous allons essentiellement apprendre à ''démarrer dBASE II''. Auparavant, si vous ne l'avez déjà fait, il est nécessaire de réaliser des copies de sécurité afin d'éviter d'avoir par la suite à utiliser la disquette originale.

1. POUR FAIRE DES COPIES DE SAUVEGARDE

Nous vous rappelons brièvement une des façons de procéder pour effectuer une copie de la face contenant le logiciel.

Commencez par ''booter'' (démarrer) à partir de la disquette système CP/M.

Formatez alors une disquette vierge à l'aide du programme :

DISCKIT3	pour le CPC 6128
DISCKIT	pour le PCW (8256 ou 8512)

(nous vous rappelons que pour exécuter un programme à partir de CP/M, il suffit de taper simplement son nom lorsque le système vous annonce qu'il attend vos ordres en affichant : **A**>)

Notez que si vous ne disposez que d'un lecteur de disquettes, il sera nécessaire d'effectuer des ''manipulations de disquettes'' appropriées. Quoiqu'il en soit, à la fin du formatage, il faudra replacer votre disquette système en A.

Recopiez alors tout le contenu de votre disquette dBASE II à l'aide du programme PIP. Pour celà, lancez d'abord ce programme par :

A>PIP

Lorsque le programme est prêt, il se manifeste en affichant *. Placez alors votre disquette dBASE II en A (si vous avez deux lecteurs, vous pouvez également placer votre disquette vierge en B). Puis frappez à la suite de * :

B:=A:*.*

Cela signifie : recopier sur B *tous* les fichiers figurant en A.

Si vous ne disposez que d'un lecteur, le programme PIP vous indiquera les changements de disquettes à effectuer (n'oubliez pas qu'alors B correspond à la disquette que vous venez de formater, tandis que A correspond à la disquette dBASE II).

Lorsque la copie est achevée, PIP vous affiche à nouveau *. Terminez en frappant ''RETURN''.

Placez alors votre nouvelle copie de dBASE II en A et vérifiez par la commande DIR qu'elle comporte bien les mêmes fichiers que la disquette originale.

Nous vous conseillons de réaliser au moins deux de ces copies. N'oubliez pas cependant que celles-ci sont strictement réservées à votre usage personnel.

2. POUR ABORDER L'ÉTUDE DE CE MANUEL

Pour vous simplifier les premières manipulations, nous avons choisi de vous faire travailler avec une seule disquette ; celle-ci devra donc contenir à la fois le logiciel dBASE II et vos fichiers de données.

Afin d'éviter de rencontrer ultérieurement des problèmes de ''manque d'espace'' sur la disquette, nous vous proposons d'utiliser une des copies précédentes sur laquelle vous ''supprimerez'' le fichier

nommé DBASEMSG.TXT. Il vous suffit pour cela d'utiliser la commande CP/M :

ERA DBASEMSG.TXT

Ne craignez pas que ce fichier vous fasse défaut par la suite. Il s'agit d'un ''petit manuel dBASE II'' que vous pourrez toujours consulter (ou lister) à partir d'une autre de vos copies.

D'autre part, pour clarifier les choses, nous vous proposons de supprimer également le petit fichier nommé NAMES.DBF par :

ERA NAMES.DBF

Remarque

Si vous disposez de deux lecteurs, il se peut que vous préfériez aborder l'étude de ce manuel en utilisant les deux avec :

— la disquette programme en A

— une disquette réservée aux ''données'' en B.

Dans ce cas, vous pouvez travailler avec une copie complète de la disquette originale (sans supprimer les deux fichiers indiqués). D'autre part, il vous faudra formater une disquette vierge supplémentaire pour y accueillir vos futurs fichiers. Enfin, lors de l'utilisation du logiciel dBASE II, il vous faudra ''mentionner'' que vos fichiers se trouvent en B, en nom en A ; nous vous indiquerons comment au moment opportun.

3. POUR DÉMARRER dBASE II

Il vous faut d'abord démarrer à partir de la disquette CP/M. Ensuite de quoi, vous remplacez la disquette système par la ''disquette de travail'' que nous venons de créer. Vous demandez alors l'exécution du programme nommé DBASE en tapant simplement ce nom (en majuscules ou en minuscules) à la suite du A>.

Vous constatez qu'alors le programme vous demande la ''date''. Certes, pour l'instant, vous ne devez guère voir l'intérêt d'une telle question. En fait, il faut savoir que Dbase conserve dans vos fichiers, soit leur date de création, soit leur dernière date de mise à jour. Si vous souhaitez que cette information ait une signification, il est souhaitable de fournir la date exacte à Dbase, à chaque fois que vous le lancez.

Votre écran doit alors se présenter comme ceci (ici, nous étions le 6 juin 1986).

```
A>dbase

* ENTREZ LA DATE SOUS LA FORME SUIVANTE (SINON RETURN)
 (JJ/MM/AA) :06/06/86

Copyright (C) 1982 RSP Inc.

***  dBASE II      Ver 2.4  1 Avril, 1983

 Frappez 'HELP', 'HELP dBASE', ou 'HELP <commande>'

 .
```

écran 2.1 : démarrage du programme DBASE

Le point qui apparaît en début de ligne est la façon dont Dbase vous signale qu'il est prêt à travailler pour vous. Dès le prochain chapitre, nous allons apprendre à lui fournir nos directives sous forme de ce qu'on appelle des *''commandes''*.

Pour l'instant, nous allons nous contenter de frapper une commande d'arrêt :

QUIT

(vous pouvez utiliser indifféremment les majuscules ou les minuscules).

Un message de fin de traitement apparaît et vous vous retrouvez à nouveau ''sous système'' (A>). Votre écran se présente maintenant ainsi :

```
A>dbase

* ENTREZ LA DATE SOUS LA FORME SUIVANTE (SINON RETURN)
 (JJ/MM/AA) :06/06/86

Copyright (C) 1982 RSP Inc.

***  dBASE II      Ver 2.4  1 Avril, 1983

 Frappez 'HELP', 'HELP dBASE', ou 'HELP <commande>'

. quit
*** FIN DU TRAITEMENT dBASE II version L.C.E ***
A>
```

écran 2.2 : démarrage et arrêt du programme DBASE

4. A PROPOS DES ÉCRANS QUE NOUS VOUS PROPOSONS

Dans ce livre nous serons amenés à vous présenter de nombreux ''écrans''. Ils vous fournissent une ''copie conforme'' de ce que vous devrez voir sur votre écran lorsque vous exécuterez les manipulations que nous décrivons.

Sachez que nous y avons profité de ce que Dbase accepte qu'on lui parle (ou qu'on lui écrive !) indifféremment en majuscules ou en minuscules. Ceci est vrai aussi bien pour les commandes que pour les réponses aux questions qu'il peut nous poser. Nous avons choisi de nous exprimer en minuscules. Comme Dbase s'exprime en majuscules, il devrait ainsi vous être assez facile de reconnaître ce qui s'affiche à l'écran de ce qu'il vous faut taper au clavier.

Notez que, par contre, dans le texte proprement dit, nous écrivons généralement les commandes en majuscules, ceci afin de mieux les mettre en évidence.

III Créons notre premier fichier

Pour exploiter des informations, il faut en disposer. Avant de pouvoir utiliser un fichier, il faut commencer par le créer. Ce chapitre va vous montrer comment.

Pour vous faire la main, nous vous proposons de créer un petit fichier du type ''répertoire téléphonique''. Nous y placerons pour chaque personne les informations suivantes :

- Nom
- Prénom
- Code postal
- Ville
- Numéro de téléphone
- Taille

Nous n'avons pas fait figurer l'adresse pour des questions de mise en page (les lignes des listes auraient été trop longues). D'autre part, la taille est une information que l'on rencontre rarement dans un tel fichier. Nous l'avons cependant choisie car elle nous permettra de montrer comment traiter une information de type ''numérique''.

1. COMMENT CRÉER UN FICHIER EN Dbase

L'opération se fait à l'aide de la commande CREATE(*). Elle se déroule en *deux étapes.* Dans un premier temps, nous définissons ce qu'on appelle la *''structure''* commune à tous les enregistrements du fichier. Dans un deuxième temps, nous procédons à la ''saisie'' proprement dite ; autrement dit, nous fabriquons un certain nombre d'enregistrements en tapant au clavier lés informations que nous souhaitons y voir figurer.

Pour vous familiariser avec la commande CREATE et les manipulations de touches qui lui sont associées, nous allons commencer par apprendre à parcourir la première étape : définition de la structure.

2. POUR DÉFINIR LA STRUCTURE D'UN FICHIER

Nous supposons que nous avons ''lancé'' le programme Dbase comme nous avons appris à le faire et en lui fournissant la date du jour. A la suite du point (qui nous montre que Dbase attend une commande), nous tapons (indifféremment en majuscules ou en minuscules) :

CREATE

Aussitôt (voyez l'écran 3.1), Dbase nous affiche : ''*** DONNEZ LE NOM DU FICHIER :'' et attend notre réponse. Ici, nous choisissons : REPERT(**) (suivi de ''RETURN''). Nous voyons alors apparaître :

DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP NOM, TYPE, DIMENSION, DECIMALE(S)

A ce niveau, il nous faut préciser les différentes rubriques (ou champs) que nous souhaitons voir figurer dans notre fichier, en leur attribuant un *nom,* un *''type''*, une *dimension* (ou taille, ou longueur) et éventuellement un *nombre de décimales* :

- *Le nom* correspond aux titres de rubriques que l'on trouverait généralement sur des fiches manuelles. Il servira ultérieurement à désigner ces différentes rubriques ; il doit donc être choisi avec soin de manière à évoquer correctement ce qu'il contient et afin de vous faciliter sa ''mémorisation''.

(*) En anglais : créer.

(**) Si vous utilisez deux lecteurs et que vous souhaitez placer votre fichier sur B, il faudra taper B:REPERT comme nom de fichier.

- *Le type* peut être **N** pour numérique ou **C** pour caractère (il existe également L pour logique mais ne l'utiliserons pas ici). Comme son nom l'indique, le type N est réservé à des informations sur lesquelles vous risquez d'être amenés à effectuer des calculs.
- *La dimension* correspond à la taille que Dbase réservera pour ce champ dans tous les enregistrements du fichier. Elle correspondra au *nombre maximum* de caractères qu'il vous sera possible d'y introduire.
- *Le nombre de décimales* peut être précisé pour les champs de type numérique.

Voyez (écran 3.1) les choix que nous avons fait. Notez que, bien que le code postal soit formé de cinq chiffres, nous l'avons prévu de type caractère car nous n'aurons jamais à effectuer de calculs sur cette information. Il en va de même pour le numéro de téléphone. Par contre, en ce qui concerne la "taille", il est intéressant de lui attribuer le type numérique. Vous voyez qu'alors il est nécessaire de définir un nombre de décimales. Ici, nous avons prévu de l'exprimer en mètres, avec une précision de 1 cm, ce qui correspond donc à 2 décimales.

```
create
*** DONNEZ LE NOM DU FICHIER : repert
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP     NOM,TYPE,DIMENSION,DECIMALE(S)
 001      nom,c,10
 002      prenom,c,8
 003      codep,c,5
 004      ville,c,8
 005      tele,c,11
 006      taille,n,4,2
 007
*** VOULEZ-COMMENCER LA SAISIE (Y/N) ? N
```

écran 3.1 : définition de la structure du fichier REPERT

(n'oubliez pas que nous nous "exprimons" en minuscules)

Nous entrons donc les descriptions de chaque champ, à raison d'une ligne par champ (donc en frappant "RETURN" à la fin de chaque ligne). Notez qu'il est toujours possible de corriger la ligne que vous êtes en train de composer. Il est, par contre, impossible de modifier d'éventuelles lignes précédentes. Nous verrons plus loin comment faire en cas d'erreur.

Bien entendu, après que nous ayons décrit le sixième champ, Dbase en attend un septième. En effet, rien ne lui dit que nous avons ter-

miné. Pour ce faire, il nous suffit de donner une description ''vide'', autrement dit de taper directement ''RETURN'', alors qu'il nous affiche 007.

Cela termine la première étape et Dbase nous demande alors si nous souhaitons effectuer la saisie ''dans la foulée'' d'un certain nombre d'enregistrements. Ici (exceptionnellement), nous répondrons N. L'exécution de la commande CREATE est achevée. Dbase signale qu'il est de nouveau à notre disposition en affichant un point.

3. POUR CONNAÎTRE LES FICHIERS EXISTANTS : LIST FILES

La commande :

LIST FILES (*)

permet d'obtenir la liste des fichiers présents sur notre disquette, le nombre d'enregistrements qu'ils contiennent et la date à laquelle ils ont été mis à jour pour la dernière fois. Dans le cas présent, voici ce qu'elle vous fournit (**) :

```
. list files

NOMS FICHIERS     ENREG.     MIS A JOUR
REPERT    DBF     00000      06/06/86
```

écran 3.2 : liste des fichiers de type DBF

Nous constatons plusieurs choses :

1) Bien que nous n'ayons encore ''saisi'' aucune information pour notre fichier REPERT, son nom apparaît ; il est toutefois mentionné comme comportant zéro enregistrement.

2) Le sigle ''DBF'' placé à droite du nom du fichier correspond à ce qu'on nomme ''extension'' ou ''type'' du fichier. Ici, cette extension est attribuée d'office par Dbase. Elle signifie ''Data Base For-

(*) En anglais : Lister les fichiers.

(**) Ce que nous décrivons ici correspond au cas où vous travaillez avec une seule disquette placée en A (même si vous disposez de deux lecteurs). Si, par contre, vous utilisez deux disquettes (A et B sur un même lecteur ou sur deux lecteurs différents), voyez également la remarque en fin de paragraphe.

mat'' (Format base de données). Nous verrons que, même sous Dbase, il existe d'autres types de fichiers.

3) Bien que notre disquette comporte d'autres fichiers (par exemple le programme DBASE lui-même, ceux-ci n'apparaissent pas ici. En fait, telle que nous l'avons utilisée, la commande LIST FILES ne considère que les fichiers possédant l'extension DBF. Nous verrons plus tard, comment connaître les autres fichiers.

Remarque : *si vous utilisez deux lecteurs (ou artificiellement deux disquettes sur un même lecteur)* :

La commande LIST FILES vous fournira les seuls fichiers de type DBF de la disquette A (probablement le fichier d'origine NAMES si vous ne l'avez pas effacé). Pour obtenir les fichiers se trouvant sur B, il faudra utiliser la commande :

LIST FILES ON B :

Notez qu'il vous est également possible de signaler à Dbase que vous souhaitez travailler avec la disquette B par la commande :

SET DEFAULT TO B

Dans ce cas, tout nom de fichier ne comportant pas d'indication de ''lecteur'' (A:, B:) sera considéré d'office comme situé sur B. De même, sans spécification explicite de votre part, les commandes se réfèreront à B. Autrement dit, en tapant cette commande après avoir lancé Dbase, il vous aurait suffit de répondre REPERT comme nom de fichier pour qu'il soit placé sur B. De même, la commande LIST FILES vous aurait affiché les fichiers contenus en B.

4. QUELQUES RÈGLES A RESPECTER

a) Les **noms de fichiers** sont formés de un à huit caractères « alphanumériques » (lettres ou chiffres), le premier devant obligatoirement être une lettre. Le blanc (ou espace) n'est pas considéré comme caractère alphanumérique. Les noms suivants :

CLIENTS VENTES12 COMMAND

sont des noms corrects. Ceux-ci ne le sont pas :

12VENTES	(le premier caractère n'est pas une lettre)
VEN+STOC	(le + n'est pas alphanumérique)
VENT JAN	(l'espace n'est pas admis)

Dbase ne distinguera pas les majuscules des minuscules et, en fait, le nom sera automatiquement converti en majuscules (Évitez les caractères accentués qui seraient ''bizarement'' convertis).

b) Les **noms de champ** sont formés suivant les mêmes règles que les noms de fichiers, mais ils peuvent comporter jusqu'à 10 caractères. Les caractères accentués sont rejetés par Dbase. Par contre le caractère ''::'' (deux points) est autorisé.

c) La **taille** de chaque champ est limitée à 254 caractères ; le nombre total de champs d'un même fichier ne peut dépasser 32. La taille totale d'un enregistrement ne peut excéder 1 000 caractères.

d) Les **champs de type caractère** peuvent renfermer une suite de caractères quelconques. Un caractère correspond à n'importe quel symbole que vous pouvez frapper à partir du clavier et afficher à l'écran. On y trouve bien entendu les lettres majuscules, les lettres minuscules, et les chiffres, mais aussi nos caractères dits ''nationaux'' :

é, è, à, ù ç ê û, ...

ainsi que d'autres caractères tels que, par exemple :

= > < & ( ) + – * / : , ; . ?

c) Les **champs de type numérique** ne peuvent contenir que des nombres décimaux avec ou sans signe (+ ou –). La virgule y est remplacée par un point et le nombre de décimales est impérativement fixé lors de la création. Le point est compté dans la taille du champ.

5. QUAND dBASE II NE RECONNAÎT PAS VOTRE COMMANDE

Lorsque vous tapez une commande que Dbase ne reconnaît pas, il vous le signale par un message comme dans cet exemple :

```
. crete
*** CETTE COMMANDE NE PEUT ETRE UTILISEE ***
crete
*** DESIREZ-VOUS CORRIGER ET RECOMMENCER (Y/N) ? N
```

écran 3.3 : en cas de mauvaise commande

Mais, en outre, il vous demande, non pas directement de frapper votre commande, mais si vous souhaitez apporter une correction à ce que vous avez tapé. Ce procédé présente surtout de l'intérêt pour les commandes un peu longues. Pour l'instant, nous vous conseillons donc de taper N (pour No, c'est-à-dire non) et Dbase vous présentera à nouveau son ''point'' d'attente.

Notez bien que cette question n'accepte comme réponse que Y (pour Yes, c'est-à-dire oui) ou N. La frappe de n'importe quelle autre touche amène Dbase à vous afficher un autre ''?'' sur la ligne inférieure.

Remarque :

D'autres messages d'erreurs peuvent être affichés par Dbase, en particulier : *** ERREUR DE SYNTAXE ***. Nous y reviendrons.

▷ **Entraînez-vous**

- Utilisez la commande CREATE pour créer la structure suivante pour un fichier nommé ESSAI :

nom de champ	Type	Longueur	Décimales
PRODUIT	C	20	
CAT	C	1	
REF	C	3	
PRIX	N	8	2
QTE	N	5	0

- Vérifiez l'existence de ce fichier.
- Voyez comment Dbase accepte de créer à nouveau une structure de *même nom,* en exécutant à nouveau cette commande CREATE avec une structure différente (par exemple, sans la rubrique PRODUIT).
- Vérifiez qu'un seul fichier ESSAI apparaît alors (*).

6. POUR FAIRE DU ''MÉNAGE'' : DELETE FILE (**)

Lorsqu'un fichier est devenu inutile, il est facile (peut-être trop !) de le supprimer à l'aide de la commande DELETE FILE suivie du nom du fichier à supprimer.

(*) Cette manipulation vous montre qu'aucun avertissement ne vous est fournit par Dbase pour vous signaler que vous allez ''écraser'' un fichier existant.

(**) En anglais : effacer fichier.

```
. delete file repert
*** LE FICHIER A ETE EFFACE  ***
```

écran 3.4 : suppression du fichier REPERT

▷ **Entraînez-vous**

- Voyez comment réagit Dbase lorsque vous cherchez à effacer un fichier qui n'existe pas.
- Voyez ce qui se passe lorsque vous essayez de créer un fichier qui existe déjà (choisissez le nom ESSAI). Entrez à nouveau la structure comme le demande Dbase.
- Voyez ce qui se passe lorsque vous cherchez à créer une "structure vide", c'est-à-dire en tapant "RETURN" alors que Dbase attend la description du premier champ (choisissez ici un nom de fichier inexistant, par exemple VIDE). Vérifiez par LIST FILES qu'aucun fichier n'a été créé.
- Voyez comment réagit Dbase lorsque vous cherchez à créer un fichier dont la structure comporte deux noms de champ identiques.
- Effacez **tous** les fichiers que nous avons fait créer jusqu'ici.

7. COMMENT REMPLIR UN FICHIER

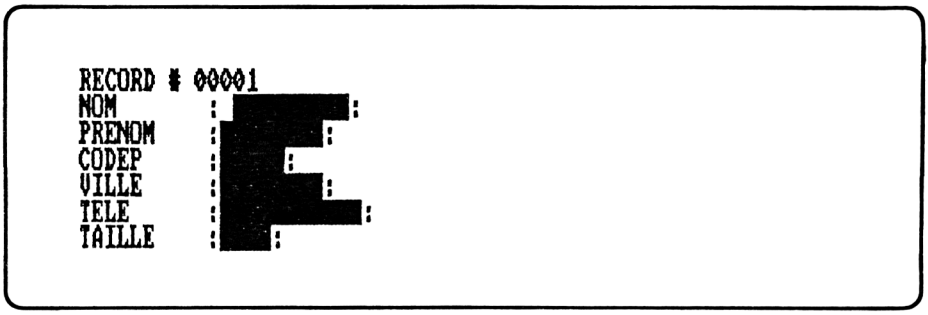
Nous vous avions indiqué au paragraphe 1 que la commande CREATE travaillait en deux étapes. Nous avons appris à réaliser la première : définir la structure. Nous allons maintenant étudier la deuxième étape : la saisie. Celle-ci ne peut-être réalisée indépendamment de la première (du moins avec la commande CREATE). C'est pourquoi nous vous proposons de créer à nouveau la structure du fichier REPERT, comme nous l'avions fait en page 13 (si vous avez conservé ce fichier jusqu'ici, effacez-le avant d'exécuter la commande CREATE).

Cette fois, à la question :

*** VOULEZ-VOUS COMMENCER LA SAISIE (Y/N) ?

nous répondons Y pour Yes (c'est-à-dire oui).

L'écran s'efface alors, puis nous voyons apparaître :



écran 3.5 : le masque de saisie d'un enregistrement

Dbase vous présente un ''masque de saisie'' pour le premier enregistrement du fichier. Vous y voyez apparaître les noms de champ et, en regard de chacun, une zone en ''inversion vidéo'' dans laquelle vous allez entrer votre information. Sa taille correspond au nombre de caractères que vous lui avez attribués lors de la définition de la structure du fichier.

Vous entrez vos informations en tenant compte des remarques suivantes :

- vous passez d'un champ au suivant, soit lorsque vous en avez rempli tous les caractères (Dbase émet alors un son), soit plus usuellement en tapant ''RETURN''.
- lorsque vous arrivez à la fin du dernier champ d'un enregistrement, Dbase efface l'écran et vous affiche le masque de saisie pour l'enregistrement suivant.
- Pour terminer la saisie, il vous suffit de laisser Dbase vous présenter le masque de saisie de l'enregistrement suivant et de taper directement ''RETURN''

Remarques

1) Dans les champs de type caractère, vous pouvez placer n'importe quels caractères de votre choix. En particulier, vous pouvez utiliser à la fois des majuscules et des minuscules. Ici, Dbase ne fera (heureusement) aucune conversion automatique.

2) Dans les champs de type numérique, il faut prendre garde à taper un ''point'' (écriture anglo-saxonne) et non une virgule !

▷ **Entraînez-vous**

- Entrez dans le fichier REPERT (dont nous venons de recréer la structure) les informations suivantes :

Nom	Prénom	Code postal	Ville	Téléphone	Taille
Thomas	Michel	58000	NEVERS	86 48 23 37	1.75
Mitenne	Laurence	83600	FREJUS	89 55 66 89	1.58
Loiseau	Albert	39400	MORREZ	84 61 28 42	1.71

(Notez que nous avons prévu un espace entre chaque groupe de deux chiffres du numéro de téléphone).

N'oubliez pas que, pour terminer votre saisie, il vous faudra taper "RETURN" quand Dbase attendra les informations correspondant au quatrième enregistrement :

- Vérifiez, à l'aide de la commande LIST FILES que votre fichier comporte effectivement 3 enregistrements.

Note : ne vous inquiétez pas trop des éventuelles "erreurs de saisie" que vous aurez pu faire. Nous verrons bientôt comment y remédier.

Avant d'aborber l'étude du chapitre suivant, nous vous conseillons de quitter Dbase par la commande QUIT.

IV Pour voir ce que nous avons créé

Jusqu'ici, nous nous sommes contentés de créer une structure de fichier et d'en saisir quelques informations. Les seules vérifications possibles se bornaient à l'examen du nom des fichiers présents sur la disquette. Nous allons maintenant apprendre à ''voir ce que nous avons créé'', aussi bien en ce qui concerne les informations elles-mêmes que la structure des fichiers. Cela va nous amener à introduire une commande fondamentale de Dbase, à savoir ''USE'' et la notion de ''zone de travail'' qui lui est associée.

1. POUR CONNAÎTRE LE CONTENU D'UN FICHIER : LIST

Dans le chapitre précédent, nous avons créé un (petit) fichier nommé REPERT comportant trois enregistrements. Nous pouvons souhaiter, tout naturellement, voir (ou revoir) le contenu de ces enregistrements. C'est le rôle de la commande LIST (*).

Comme nous vous l'avions demandé à la fin du précédent chapitre,

(*) En anglais : faire la liste (lister)

nous supposons que nous avions quitté Dbase. Nous le lançons donc à nouveau (A > DBASE). Puis nous indiquons que nous souhaitons "utiliser" (use en Anglais) le fichier "repert", en tapant la commande :

USE REPERT

Notez qu'aucune réponse particulière (autre que le point) ne nous est fournie. La commande LIST nous permet alors d'afficher le contenu de notre fichier :

```
A>dbase

* ENTREZ LA DATE SOUS LA FORME SUIVANTE (SINON RETURN)
 (JJ/MM/AA) :07/06/86

Copyright (C) 1982 RSP Inc.

***  dBASE II      Ver 2.4  1 Avril, 1983

 Frappez 'HELP', 'HELP dBASE', ou 'HELP <commande>'

. use repert
. list
00001  Thomas     Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert    39400 MORREZ   84 61 28 42  1.71
```

écran 4.1 : pour connaître le contenu d'un fichier : LIST

Remarques

1) La commande USE établit un lien avec un fichier existant. Ce lien subsiste tant qu'on n'en crée par un nouveau (par exemple par une nouvelle commande USE). Nous illustrerons ce point, dans le paragraphe 7.

2) La commande LIST ne nous précise pas les noms des différents champs dont elle nous affiche le contenu.

3) La commande LIST nous fournit un "numéro d'enregistrement" (ici 1, 2 et 3). Si vous ne souhaitez pas le voir apparaître, il vous suffit de remplacer LIST par LIST OFF :

```
. list off
Thomas      Michel    58000 NEVERS    86 48 23 37   1.75
Mitenne     Laurence  83600 FREJUS    89 55 66 89   1.58
Loiseau     Albert    39400 MORREZ    84 61 28 42   1.71
```

écran 4.2 : pour "lister" sans les numéros d'enregistrement : LIST OFF

2. POUR RETROUVER LA STRUCTURE D'UN FICHIER : LIST STRUCTURE

Il peut être utile de retrouver la structure exacte d'un fichier, par exemple :

- pour retrouver les noms des champs, leur type ou leur longueur. Celà sera véritablement indispensable lorsque nous effectuerons des recherches sur nos fichiers.
- pour s'assurer que notre définition de structure a été convenablement interprétée par Dbase lors de la première étape de la commande CREATE. A ce niveau, en effet, certaines « maladresses » n'entraînent aucun message d'avertissement.

Ce sera le rôle de la commande :

LIST STRUCTURE

Voici, à titre d'exemple, comment nous retrouvons la structure de notre fichier REPERT :

```
. list structure
STRUCTURE DU FICHIER                : A:REPERT   .DBF
NOMBRE D'ENREGISTREMENTS            : 00003
DATE DE LA DERNIERE MISE A JOUR     : 06/06/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP    NOM          TYP    DIM     DECIMALE(S)
001      NOM           C     010
002      PRENOM        C     008
003      CODEP         C     005
004      VILLE         C     008
005      TELE          C     011
006      TAILLE        N     004     002

** TOTAL **                 00047
```

écran 4.3 : pour afficher la structure d'un fichier : LIST STRUCTURE

Vous notez tout d'abord que Dbase nous rappelle le nom du fichier avec lequel nous travaillons. Ici, son nom est "préfixé" par **A** : car nous travaillons (par défaut) avec cette unité de disquette (*). Par ailleurs la quatrième ligne : "BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION" peut vous paraître assez obscure. Ne cherchez pas à l'interpréter pour l'instant ; cela s'éclairera plus tard (**).

3. NOTION DE "ZONE DE TRAVAIL"

La commande USE établit en quelque sorte un "lien" avec un fichier donné. En même temps, elle réserve en mémoire centrale une "zone de travail" attribuée au fichier en question. Plus précisément, une partie ou la totalité du fichier (suivant sa taille) sera recopiée dans cette zone, de manière à accélérer le traitement(***). Bien entendu, lorsque vous serez amenés à effectuer des modifications de certains enregistrements, celles-ci seront réalisées d'abord dans la zone de travail ; mais celle-ci sera recopiée sur disquette dès que nécessaire. Ce qui montre l'intérêt qu'il y a à quitter convenablement Dbase (par QUIT). Couper immédiatement votre Amstrad risquerait d'empêcher que certaines modifications ne soient reportées sur disquette.

4. TOUTES LES COMMANDES NE PORTENT PAS SUR LA "ZONE DE TRAVAIL"

Certaines commandes portent directement sur le fichier associé par USE à la zone de travail. C'est le cas, par exemple, de LIST, LIST STRUCTURE. Vous pouvez d'ailleurs remarquer que lorsque vous lancez de telles commandes, le lecteur de disquette ne se met pas en route (du moins tant que le fichier concerné n'est pas trop gros) ; cela montre bien que l'information recherchée se trouve déjà en mémoire centrale.

(*) Si vous utilisez deux disquettes et que, au préalable, vous avez exécuté la commande SET DEFAULT TO B:, le nom sera "préfixé" par B:.

(**) Sachez qu'en Dbase II, il existe en fait deux zones actives : l'une nommée primaire, l'autre nommée secondaire.

(***) En jargon d'informaticien, cet ensemble d'opérations porte le nom d'ouverture de fichier.

D'autres commandes, par contre, sont totalement indépendantes de la zone de travail. Ainsi, par exemple, LIST FILES va examiner directement le contenu de la disquette. Il est alors important de noter que de telles commandes n'affectent aucunement le contenu de la zone de travail et donc le lien établi par USE.

Voici un exemple illustrant ce que nous venons de dire :

```
use repert
. list
00001  Thomas     Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert    39400 MORREZ   84 61 28 42  1.71
. list files

NOMS FICHIERS    ENREG.     MIS A JOUR
REPERT   DBF     00003      06/06/86


. list structure
STRUCTURE DU FICHIER              : A:REPERT   .DBF
NOMBRE D'ENREGISTREMENTS          : 00003
DATE DE LA DERNIERE MISE A JOUR : 06/06/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP   NOM          TYP   DIM    DECIMALE(S)
001     NOM           C    010
002     PRENOM        C    008
003     CODEP         C    005
004     VILLE         C    008
005     TELE          C    011
006     TAILLE        N    004    002

** TOTAL **              00047
. delete file repert
*** FICHIER EN COURS D'UTILISATION ***
```

écran 4.4 : panachage de commandes utilisant ou n'utilisant pas la "zone de travail"

Vous notez que LIST FILES s'exécute sans problème. Le résultat de la commande LIST STRUCTURE montre clairement que le lien avec le fichier REPERT existe toujours. Par contre, vous remarquez que Dbase refuse une commande d'effacement du fichier correspondant à la zone de travail. Par contre, DELETE appliqué à tout autre fichier que REPERT n'aurait posé aucun problème.

5. *QUAND AUCUN LIEN N'A ÉTÉ ÉTABLI*

Si vous demandez d'exécuter une commande qui utilise la zone de travail, alors qu'aucun lien n'a été établi, Dbase vous le signale clairement. Il vous propose alors d'établir ce lien en vous demandant de lui fournir le nom du fichier concerné. En voici un exemple :

```
A>dbase

* ENTREZ LA DATE SOUS LA FORME SUIVANTE (SINON RETURN)
 (JJ/MM/AA) :08/06/86

Copyright (C) 1982 RSP Inc.

***  dBASE II      Ver 2.4  1 Avril, 1983

 Frappez 'HELP', 'HELP dBASE', ou 'HELP <commande>'

. list
*** AUCUNE BASE N'EST UTILISEE, DONNEZ SON NOM : repert
00001  Thomas     Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert    39400 MORREZ   84 61 28 42  1.71
. list off
Thomas     Michel    58000 NEVERS   86 48 23 37  1.75
Mitenne    Laurence  83600 FREJUS   89 55 66 89  1.58
Loiseau    Albert    39400 MORREZ   84 61 28 42  1.71
```

écran 4.5 : quand on oublie la commande USE

Vous notez que, grâce à la question complémentaire posée par Dbase, tout se passe exactement comme si vous aviez exécuté la commande :

USE REPERT

Le cas particulier de CREATE

Nous avons utilisé CREATE sans avoir établi au préalable un lien par USE. Celà est, somme toute, assez compréhensible dans la mesure où USE concerne un *fichier existant*.

Par contre, à la fin de la création d'un fichier, nous pourions penser qu'un lien a été établi avec ce fichier. Mais ce n'est pas le cas.

D'autre part, si vous exécutez une commande CREATE après avoir établi un lien avec un fichier, vous constaterez que ce lien se trouve détruit.

En résumé, CREATE supprime l'éventuel lien existant, mais n'en crée pas de nouveau.

▷ **Entraînez-vous**

- Assurez-vous que vous possédez toujours le fichier REPERT, tel qu'il apparaît en page . (Si ce n'est pas le cas, créez-le à nouveau).
- Créez un second fichier nommé STOCK, ayant la même structure que le fichier ESSAI (voir page et comportant les informations suivantes (*) :

PRODUIT	CAT	REF	PRIX	QTE
Cafetière 12 T	A	432	137.47	32
Four à micro-ondes	E	248	1 875.25	7
Grille pain	A	521	158.75	23
Four à raclette 6 P	A	427	133.72	15

- Tapez LIST. Voyez comme Dbase vous demande un nom de fichier. Répondez REPERT.
- Vérifiez que le lien avec REPERT a été convenablement établi en tapant la commande :
  - LIST STRUCTURE
- Tapez maintenant successivement ces deux commandes :
  - USE STOCK
  - LIST
- Tapez alors :
  - USE
  - LIST

(répondez simplement en tapant "RETURN" à la question posée par Dbase).

6. UN SEUL LIEN A LA FOIS

Les manipulations proposées ci-dessus vous ont montré que vous ne pouvez établir de lien qu'avec un seul fichier à la fois. En quelque sorte, "tout nouveau lien chasse l'ancien". En outre, USE (tout court) supprime l'éventuel lien existant.

(*) Ici, vous pouvez utiliser sans crainte nos caractères nationaux.

Exceptionnellement, nous vous proposons un écran correspondant aux manipulations demandées dans ''Entraînez-vous'', à l'exception de la phase de création du fichier STOCK (symbolisée par des ''points'').

```
. create
*** DONNEZ LE NOM DU FICHIER : stock
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP     NOM,TYPE,DIMENSION,DECIMALE(S)
 001      produit,c,20
 002      cat,c,1
 003      ref,c,3
 004      prix,n,8,2
 005      qte,n,5
 006
*** VOULEZ-COMMENCER LA SAISIE (Y/N) ? Y

                 •
                 •
                 •
                 •
                 •

. list
*** AUCUNE BASE N'EST UTILISEE, DONNEZ SON NOM : repert
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
. list structure
STRUCTURE DU FICHIER              : A:REPERT  .DBF
NOMBRE D'ENREGISTREMENTS          : 00003
DATE DE LA DERNIERE MISE A JOUR : 08/06/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP    NOM          TYP   DIM    DECIMALE(S)
001      NOM           C    010
002      PRENOM        C    008
003      CODEP         C    005
004      VILLE         C    008
005      TELE          C    011
006      TAILLE        N    004    002

** TOTAL **              00047


. use stock
. list
00001  Cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
. use
. list
*** AUCUNE BASE N'EST UTILISEE, DONNEZ SON NOM :
```

écran 4.6 : comment les liens se font et se défont

7. POUR OBTENIR DES INFORMATIONS PERMANENTES : UTILISER L'IMPRIMANTE

Les informations affichées à l'écran ont un caractère fugitif ; dans certains cas vous pouvez souhaiter en conserver une trace écrite à l'aide d'une imprimante. Il existe plusieurs façons d'imprimer à partir de Dbase. Pour l'instant nous vous proposons la plus simple, à savoir utiliser les touches :

Alt/P (*) (sur PCW) **Ctrl/P** (**) (sur CPC)

Comme ''sous le système CP/M'', vous pouvez frapper la combinaison de touches Alt/P pour demander que tout ce qui s'affiche à l'écran soit *également* envoyé à l'imprimante. Pour interrompre l'impression, il suffit de presser à nouveau Alt/P.

En Dbase, il est cependant déconseillé d'utiliser cette possibilité lors de l'exécution des commandes ''plein écran'' (pour l'instant, la seule commande de ce type que nous ayons rencontrée est CREATE, mais il en existe d'autres : EDIT, APPEND...).

(*) Alt/P doit se comprendre ainsi : appuyez sur la touche Alt et, tout en gardant cette touche enfoncée, tapez P.

(**) Ctrl/P doit se comprendre ainsi : appuyez sur la touche marquée CONTROL et, tout en gardant cette touche enfoncée, tapez P.

V La mise à jour "A VUE"

Créer un fichier, c'est bien ! Encore faut-il pouvoir le "mettre à jour" en cas de besoin :

— pour ajouter de nouveaux enregistrements ou pour en supprimer.
— pour corriger d'éventuelles erreurs de saisie
— pour tenir compte de l'évolution de certaines informations.

En Dbase, la mise à jour *peut* se dérouler de manière analogue à ce que vous feriez avec un fichier "manuel", c'est-à-dire en travaillant *visuellement* sur une seule fiche à la fois. C'est à ces possibilités que nous consacrons ce chapitre. Nous découvrirons plus loin que Dbase dispose d'autres méthodes de mise à jour, plus automatiques et plus puissantes.

Si vous souhaitez suivre sur votre Amstrad les manipulations que nous décrivons, assurez-vous que vous disposez bien des deux fichiers REPERT et STOCK tels qu'ils sont listés en page 28.

1. POUR MODIFIER LE CONTENU DE CERTAINS ENREGISTREMENTS : EDIT

Pour modifier le contenu d'un enregistrement, il nous suffit d'employer la commande :

EDIT

suivie du numéro de l'enregistrement concerné. Par exemple :

- USE REPERT
- EDIT 2

vous affiche l'enregistrement numéro 2 (du fichier REPERT) en mode ''plein écran'', c'est-à-dire comme il se présentait lorsque vous l'avez fabriqué par CREATE.

Vous pouvez y effectuer les corrections de votre choix grâce à des touches (ou combinaisons de touches) appropriées. Certaines déplacent le curseur à l'intérieur d'un champ, d'autres permettent de changer de champ (voir tableau 5.2). Les corrections proprement dit se font en combinant les opérations suivantes :

— remplacement d'un caractère par un autre : il suffit de frapper un nouveau caractère qui prend la place de celui sur lequel se trouve le curseur.

— effacement du caractère situé sous le curseur : Alt/G sur PCW ou Ctrl/G sur CPC

— insertion de un ou plusieurs caractères à gauche du curseur ; il faut, pour ce faire, passer en ''mode insertion'' (par Alt/V ou Ctrl/V) ; ce mode se signale par le mot INSERT qui s'affiche en haut à droite du numéro d'enregistrement. On annule cet effet d'insertion, de la même manière qu'on l'a mis en place (Alt/V ou Ctrl/V). Notez que tant qu'on le laisse actif, il est impossible d'effectuer de simples remplacements.

Déplacement curseur	gauche	Alt/S (*)
	droite	Alt/D (*)
Changement de champ	précédent	← ou Alt/E ou Alt/A (*)
	suivant	→ ou Alt/X ou Alt/F ou ''return'' (*)
Corrections	Effacement un caractère	Alt/G (*)
	Insertion (passage et suppression)	Alt/V (*)

(*) Sur CPC, Alt sera remplacée par Ctrl.

tableau 5.1 : les touches de correction d'un enregistrement

Que faire lorsque vos corrections ont ainsi été apportées ? Dbase vous offre le choix entre :

— corriger d'autres enregistrements ; certaines touches vous font passer sur l'enregistrement suivant ou sur le précédent. Vous pouvez ainsi, par une seule commande EDIT, modifier plusieurs enregistrements (même s'ils ne sont pas consécutifs).

— En ''finir'' avec la commande EDIT. Cela est bien entendu nécessaire, que vos corrections aient porté sur un seul enregistrement ou sur plusieurs. En principe cela se fait par Alt/W. Il est également possible de terminer en demandant à Dbase d'ignorer les modifications apportées à l'enregistrement courant (Alt/Q). Notez bien que l'enregistrement courant est celui qui apparaît à l'écran lorsque vous tapez Alt/Q ; ce n'est pas nécessairement le dernier modifié !

Changement d'enregistrement	suivant	Alt/C (*)
	précédent	Alt/R (*)
Terminer les corrections	Normalement	Alt/W (*)
	En ignorant les corrections de l'enregistrement courant	Alt/Q (*)

(*) Sur CPC, Alt sera remplacé par Ctrl.

tableau 5.2 : les touches de changement d'enregistrement et de fin de corrections

Remarques :

① Lorsque le curseur est sur le dernier champ d'un enregistrement, la touche → fait également passer à l'enregistrement suivant.

② De même, lorsque le curseur est sur le premier champ, la touche ← fait passer à l'enregistrement précédent.

③ Lorsque vous êtes ''sur'' le dernier enregistrement, Alt/C (passage à l'enregistrement suivant) est équivalent à Alt/W.

④ Lorsque vous êtes ''sur'' le premier enregistrement, Alt/R (passage à l'enregistrement précédent) est lui aussi équivalent à Alt/W.

▷ **Entraînez-vous**

- Assurez-vous que vous disposez bien du fichier REPERT tel qu'il apparaît en page 28.
- Remplacez, à l'aide EDIT 1, le nom Thomas du premier enregistrement par Thamas (*remplacement* de o par a). Terminez par Alt/W et vérifiez par LIST que votre correction a été effectuée.
- Remplacez, dans l'enregistrement numéro 3, le prénom Albert par Alberte. Vérifiez.
- Remplacez le nom Mitenne par Mitene (*effacement* d'un caractère). Vérifiez.
- Remplacez le prénom Michel par Michael (*insertion* d'un caractère). Vérifiez.
- Effectuez plusieurs autres modifications de votre choix.
- Faites les modifications nécessaires pour retrouver le fichier dans son état initial.

Attention au type numérique

Sur les champs de type caractère, les modifications ne posent pas (trop) de problèmes et vous voyez en permanence à l'écran ce qui sera en définitive enregistré dans le fichier. Par contre, il n'en va plus exactement de même pour les champs numériques. En effet, si vous commencez à déplacer le curseur sur un tel champ, Dbase ne conservera en définitive que la partie sur laquelle vous avez déplacé le curseur (avec les éventuelles modifications que vous aurez apportées). Bien sûr, si vous ne faites que passer sur un champ numérique (en progressant d'un champ au suivant), aucune modification n'aura (heureusement) lieu.

2. POUR AJOUTER DES ENREGISTREMENTS EN FIN DE FICHIER : APPEND

Supposons que nous souhaitions compléter notre fichier REPERT (comportant pour l'instant trois enregistrements). Il nous suffit de taper :

- USE REPERT (si ce lien n'a pas encore été établi)
- **APPEND**

Nous voyons aussitôt apparaître, pour le quatrième enregistrement, un ''masque de saisie'' identique à celui que nous procurait CREATE. Nous pouvons ainsi ajouter autant d'enregistrements que nous le souhaitons. Pour terminer, il nous suffit de taper ''RETURN'' alors que Dbase attend des informations pour un nouvel enregistrement (en toute rigueur, Alt/W et Alt/Q sont également utilisables avec la même signification que ''sous'' EDIT — Ils le sont d'ailleurs également avec CREATE !).

▷ **Entraînez-vous**

- Ajoutez les deux enregistrements suivants au fichier REPERT :

  Meunier Albert 75008 PARIS 49 33 27 65 1.83
  Taillefert Claude 45510 SULLY 77 89 63 10 1.78

- Vérifiez par LIST.
- Ajoutez cet enregistrement :

  Grillon Jules 75006 PARIS 42 48 15 30 1.68

- Vérifiez par LIST. Votre fichier doit maintenant se présenter ainsi :

```
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
```

- Ajoutez deux enregistrements au fichier STOCK de manière à ce qu'il se présente ainsi

```
00001  Cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
00005  Friteuse 1 kg         B 433    349.25     21
00006  Table de cuisson      E 647   2450.00      6
```

3. POUR VOIR UN ENREGISTREMENT PARTICULIER

Jusqu'ici, nous avons affiché le contenu de nos fichiers à l'écran. Certes, cela ne posait guère de problèmes dans la mesure où ces fichiers étaient suffisamment petits pour que leur affichage n'occupe pas plus d'un écran. Mais, il n'en irait plus de même avec, par exemple, 500 ou 2 000 enregistrements ! La liste défilerait devant nos yeux et ne se ''figerait'' que sur la fin du fichier.

Dbase nous propose plusieurs façons de résoudre ce problème. L'une

d'entre elles consiste à se déplacer à l'intérieur du fichier en manipulant ce qu'on nomme le ''**pointeur d'enregistrement**''. Ce sera l'objet de la commande :

GOTO(*)

suivie du numéro de l'enregistrement sur lequel on souhaite placer le pointeur.

Essayez cette petite manipulation sur le fichier REPERT :

```
use repert
. goto 3
. display
00003  Loiseau     Albert    39400 MORREZ    84 61 28 42   1.71
```

écran 5.3 : manipuler le ''pointeur'' et afficher l'enregistrement courant.

La commande GOTO 3 place le pointeur sur l'enregistrement numéro 3. Mais elle ne fait que cela ; en particulier elle ne l'affiche pas. La commande :

DISPLAY(**)

affiche l'enregistrement courant, c'est-à-dire celui qui est désigné par le pointeur. Notez bien que DISPLAY ne modifie aucunement la valeur du pointeur. Pour vous en convaincre, il suffit d'exécuter deux fois de suite cette même commande : vous obtiendrez deux fois le même enregistrement.

Remarque

Ne concluez pas hâtivement que LIST concerne tout le fichier alors que DISPLAY ne concerne qu'un seul enregistrement. Nous verrons plus loin que la réalité est un peu plus complexe.

4. POUR SUPPRIMER UN ENREGISTREMENT

En Dbase, la suppression d'enregistrements se fait en deux temps totalement indépendants. Plus précisément, il existe une commande

(*) En anglais : aller à.
(**) En anglais : afficher.

(PACK) dont le but est de supprimer définitivement les enregistrements qui ont été *préalablement marqués* pour effacement. Nous allons donc apprendre comment placer une telle marque.

4.1. Marquer des enregistrements pour effacement : DELETE

La commande :

DELETE(*)

marque pour effacement *l'enregistrement courant*. En voici un exemple :

```
. use repert
. goto 3
. display
00003  Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
. delete
00001 ENREGISTREMENT(S) EFFACE(S)
. display
00003 *Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
. goto 5
. delete
00001 ENREGISTREMENT(S) EFFACE(S)
. display
00005 *Taillefert Claude     45510 SULLY     77 89 63 10  1.78
. list
00001  Thomas      Michel    58000 NEVERS    86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS    89 55 66 89  1.58
00003 *Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS     49 33 27 65  1.83
00005 *Taillefert Claude     45510 SULLY     77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS     42 48 15 30  1.63
```

écran 5.4 : Pour marquer un enregistrement pour effacement : DELETE

Notez bien comment Dbase vous signale qu'un enregistrement a été marqué pour effacement (présence de * lorsque vous en affichez le contenu). Bien entendu, vous pouvez marquer ainsi autant d'enregistrements que vous le souhaitez.

(*) En anglais : effacer.

4.2. Les supprimer définitivement

Il suffit d'exécuter la commande :

PACK(*)

Celle-ci supprime *définitivement* tous les enregistrements marqués. Notez qu'alors, les numéros des enregistrements se trouvent modifiés.

```
. use repert
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003 *Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005 *Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
. pack
COMPACTAGE TERMINE : 00004 ENREGISTREMENT(S)  TRANSFERE(S)
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00004  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
```

écran 5.5 Pour suprimer définitivement les enregistrements marqués : PACK

Remarque

Cette opération oblige Dbase à effectuer une sorte de « recopie » du fichier sur lui même. Elle peut être gourmande en temps lorsqu'elle porte sur des fichiers relativement importants. Dans ce cas, il est souhaitable de se contenter de marquer les enregistrements concernés et de ne lancer PACK qu'une fois de temps en temps.

▷ **Entraînez-vous**

- Si vous ne l'avez pas fait en lisant ce paragraphe, supprimer les enregistrements 3 et 5 du fichier REPERT (qui doit donc se présenter comme dans la dernière liste de l'écran 5.5)

(*) En anglais : compacter.

5. EN CAS DE REMORDS, POUR SUPPRIMER DES MARQUES D'EFFACEMENT : RECALL

La commande :

RECALL(*)

Supprime (si elle existe) la marque d'effacement de l'enregistrement *Courant :*

```
. use repert
. list
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00004  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. goto 3
. delete
00001 ENREGISTREMENT(S) EFFACE(S)
. goto 1
. delete
00001 ENREGISTREMENT(S) EFFACE(S)
. list
00001 *Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003 *Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00004  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. goto 3
. recall
00001 ENREGISTREMENT(S) RESTITUE(S)
. list
00001 *Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00004  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
```

écran 5.6 : Pour supprimer une marque d'effacement : RECALL

Pour ceux dont les remords sont sans bornes, la commande RECALL ALL permet de supprimer toutes les marques.

(*) En anglais : rappeler.

6. PLUSIEURS FAÇONS DE MARQUER UN ENREGISTREMENT POUR EFFACEMENT

Nous avons vu le rôle de DELETE. Mais, dans tous les modes où vous travaillez en plein écran (EDIT, CREATE, APPEND), vous pouvez mettre une marque ou supprimer une marque en frappant **alt/U** (Ctrl/U). Notez que la même combinaison de touches permet de *poser* ou de *supprimer* la marque.

Enfin, sachez que dans ces modes, la marque se manifeste, non plus par *, mais par le texte DELETED(*) qui apparait à droite du numéro d'enregistrement comme dans cet exemple :

```
RECORD # 00001                    DELETED
NOM       :Thomas      :
PRENOM    :Michel      :
CODEP     :58000:
VILLE     :NEVERS      :
TELE      :86 48 23 37:
TAILLE    :1.75:
```

écran 5.7 : Un enregistrement marqué pour effacement en mode "plein écran"

▷ **Entraînez-vous**

- Redonnez à votre fichier REPERT l'état qu'il avait à la fin de l'écran 5.5 de la page 37 par
  - GOTO 1
  - RECALL
- Marquez pour effacement, par DELETE, l'enregistrement numéro 2. Vérifiez.
- Faites : EDIT 4 et marquez l'enregistrement par Alt/U.
  Vérifiez.
- Faites EDIT 2 et enlevez la marque d'effacement de l'enregistrement numéro 2. Vérifiez.
- Marquez l'enregistrement 3 par DELETE
- Voyez ce qui se passe si vous chercher à marquer par DELETE les enregistrements 3 et 4 qui le sont déjà.
- Supprimez toutes les marques par RECALL ALL. Vérifiez (votre fichier doit se trouver dans le même état qu'au début de "Entraînez-vous").

(*) En anglais : effacé.

7. POUR INSÉRER UN ENREGISTREMENT

Très souvent, vous vous contenterez d'ajouter des enregistrements en fin de fichier (par APPEND). Si toutefois, vous souhaitez pouvoir « insérer » de nouveaux enregistrements entre d'autres, sachez que Dbase vous permet de le faire. La commande :

INSERT(*)

permet d'insérer **un** enregistrement (et un seul) **après** l'enregistrement courant. Cette commande vous propose un masque de saisie (comme APPEND). Notez bien que les numéros de tous les enregistrements suivant l'enregistrement courant sont alors augmentés de une unité.

De façon analogue, la commande

INSERT BEFORE(**)

permet d'insérer **un** enregistrement **avant** l'enregistrement courant.

▷ **Entraînez-vous**

Sur le fichier REPERT qui doit se présenter comme à la fin de l'écran 5.5 :

- Insérer les enregistrements nécessaires pour lui redonner le même contenu qu'en page...
- Voyez comme Dbase refuse que vous insériez un enregistrement après le dernier du fichier (Vous pouvez toujours utiliser APPEND).

(*) En anglais : insérer.
(**) En anglais : insérer avant.

VI Consulter un fichier

Nous savons maintenant *réaliser* un fichier et *le mettre à jour*. Par contre, en ce qui concerne sa *consultation,* nous nous sommes jusqu'ici limités à la liste globale où à celle d'un seul enregistrement.

Nous allons maintenant découvrir d'autres manières, plus riches, de tirer de l'information d'un fichier. Ainsi, nous verrons comment n'en lister qu'une partie ou ne faire apparaître que les contenus de certains champs.

Plus généralement, nous verrons comment afficher, non seulement la simple valeur d'un champ, mais celle de ce que l'on nomme une ''expression''. Cette notion d'expression est fondamentale en Dbase et nous serons amenés à l'utiliser fréquemment par la suite.

1. CONSULTATION GLOBALE OU RESTREINTE

Voyons déjà comment afficher une partie d'un fichier, ce qui va nous amener à faire quelques rappels.

1.1. Consultation globale

Nous avons déjà vu le rôle de LIST. La commande :

DISPLAY ALL(*)

fait pratiquement la même chose :

```
use repert
. display all
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
```

écran 6.1 : pour afficher le contenu d'un fichier, écran par écran : DISPLAY ALL

La seule différence entre DISPLAY ALL et LIST réside dans le fait que la première effectue l'affichage écran par écran. Autrement dit, pour les fichiers de plus de 25 lignes, l'affichage s'interrompt à chaque fois que l'écran est rempli. Il vous faut frapper sur une touche quelconque pour qu'il se poursuive. Bien entendu, c'est là le seul moyen réaliste d'examiner effectivement les enregistrements d'un fichier important.

Remarque : LIST ALL provoque la même chose que LIST.

1.2. Consultation enregistrement par enregistrement

Nous avons déjà vu comment afficher un enregistrement de numéro donné à l'aide des commandes GOTO et DISPLAY. Il est également possible d'obtenir le même résultat en spécifiant, dans la commande DISPLAY, le numéro de l'enregistrement à lister, précédé du mot RECORD(**). Par exemple :

DISPLAY RECORD 5

signifie : afficher l'enregistrement numéro 5.

(*) En anglais : afficher tout.
(**) En anglais : enregistrement.

```
. use repert
. display record 5
00005  Taillefert Claude    45510 SULLY     77 89 63 10   1.78
```

écran 6.2 : pour afficher un seul enregistrement

Remarque : LIST RECORD 5 aurait le même effet que DISPLAY RECORD 5.

1.3. Consultation restreinte à quelques enregistrements consécutifs

```
. use repert
. goto 2
. display next 3
00002  Mitenne     Laurence 83600 FREJUS    89 55 66 89   1.58
00003  Loiseau     Albert   39400 MORREZ    84 61 28 42   1.71
00004  Meunier     Albert   75008 PARIS     49 33 27 65   1.83
```

écran 6.3 : pour afficher plusieurs enregistrements consécutifs

DISPLAY NEXT 3

signifie : lister trois enregistrements consécutifs, à partir de l'enregistrement courant. (Bien entendu, le ''pointeur'' se trouve du même coup augmenté de 3).

▷ **Entraînez-vous**

- Listez les enregistrements 3 à 5 du fichier STOCK
- Voyez ce qui se passe lorsque vous demandez plus d'enregistrements qu'il n'en reste, par exemple avec :
  - GOTO 3
  - DISPLAY NEXT 10
- Voyez ce qui se passe lorsque vous cherchez à placer le pointeur « en dehors » du fichier, par exemple avec :
  - GOTO 12

1.4. En définitive

Les commandes LIST et DISPLAY peuvent toutes deux être assorties de ce que l'on nomme une ''**étendue**'', à savoir :

ALL	pour tout le fichier
RECORD 5	pour un enregistrement précis (ici le 5)
NEXT 3	pour les 3 (ici) enregistrements suivants

Les comportements de LIST et DISPLAY sont alors très voisins. La seule grande différence apparaît au niveau de ces commandes employées sans indicateur d'étendue. Ainsi :

LIST	correspond à	DISPLAY ALL
DISPLAY	correspond à	LIST NEXT 1

2. POUR N'AFFICHER QUE CERTAINS CHAMPS

Jusqu'ici, LIST et DISPLAY vous affichaient les valeurs de tous les champs (plus, d'ailleurs le numéro d'enregistrement). Vous pouvez aisément demander à Dbase de n'afficher que les champs qui vous intéressent :

```
. use repert
. list prenom,tele
00001  Michel    86 48 23 37
00002  Laurence  89 55 66 89
00003  Albert    84 61 28 42
00004  Albert    49 33 27 65
00005  Claude    77 89 63 10
00006  Jules     42 48 15 30
. goto 3
. display prenom,ville
00003  Albert    MORREZ
```

écran 6.4 : pour n'afficher que certains champs (1)

Vous pouvez également ''combiner'' avec l'indicateur d'étendue :

```
. use repert
. goto 2
. list next 3 prenom,tele
00002  Laurence  89 55 66 89
00003  Albert    84 61 28 42
00004  Albert    49 33 27 65
. display record 3 prenom,ville
00003  Albert    MORREZ
```

écran 6.5 : pour n'afficher que certains champs (2)

▷ **Entraînez-vous**

- Affichez les champs NOM et TAILLE du fichier REPERT.
- Essayez d'obtenir le même résultat, sans les numéros d'enregistrement.
- Affichez les champs contenant le nom, la ville et le code postal du fichier REPERT (si vous ne vous souvenez plus du nom de l'un d'entre eux, utilisez LIST STRUCTURE).
- Même question pour les enregistrements 2 à 4.
- Reprenez les commandes d'affichage de l'écran 6.5 en plaçant l'indicateur d'étendue après les noms de champ (au lieu de avant).
- Voyez comme il est possible d'afficher les champs dans un ordre différent de celui où ils apparaissent dans l'enregistrement. Essayez, par exemple :
  - LIST TAILLE, NOM
- Voyez comme il est possible d'afficher plusieurs fois le même champ. Essayez, par exemple :
  - LIST NOM, TAILLE, NOM
  - LIST OFF NOM, TAILLE, NOM
- Voyez comment réagit Dbase lorsque vous fournissez deux fois l'indicateur d'étendue. Essayez, par exemple :
  - LIST ALL NOM, PRENOM RECORD 5
- Voyez ce qui se passe lorsque vous n'indiquez pas tous les champs à la suite les uns des autres comme dans :
  - LIST NOM, PRENOM ALL TELEPHONE
- Voyez ce qui se passe si vous oubliez une virgule entre deux noms de champ, comme dans :
  - LIST NOM TAILLE

3. POUR AFFICHER DES "EXPRESSIONS"

Jusqu'ici, nous nous sommes contentés d'afficher directement la valeur de un ou plusieurs champs. En fait, Dbase vous permet d'afficher les valeurs de ce qu'on appelle des *"expressions"*. Il s'agit là d'une notion très générale qui peut intervenir dans beaucoup de commandes Dbase. Nous allons commencer par l'introduire sur des exemples.

```
use repert
. list nom, taille-1.70
00001  Thomas       0.05
00002  Mitenne      -.12
00003  Loiseau      0.01
00004  Meunier      0.13
00005  Taillefert   0.08
00006  Grillon      -.02
. list off prenom, 2*taille
Michel     3.50
Laurence   3.16
Albert     3.42
Albert     3.66
Claude     3.56
Jules      3.36
```

écran 6.6 : affichage d'expressions numériques (1)

L'expression :

TAILLE – 1.70

exprime la différence (notée –) entre la valeur du champ TAILLE et le nombre 1.70 (on appelle aussi ce dernier une constante numérique). Cette expression est dite numérique car elle fournit un *résultat* de ce type.

De même :

2 * TAILLE

est une expression numérique qui correspond au produit (noté * et non ×) de la valeur du champ TAILLE par le nombre 2.

```
. use stock
. list off produit, prix*qte
Cafetière 12 T              4399.04
Four à micro-ondes         13126.75
Grille pain                 3651.25
Four à raclette 6 P         2005.80
Friteuse 1 kg               7334.25
Table de cuisson           14700.00
```

écran 6.7 : affichage d'expressions numériques (2)

Ici :

PRIX * QTE

est une expression de type numérique qui demande d'effectuer le produit des valeurs des champs PRIX et QTE. Elle nous fournit ainsi, pour chaque enregistrement, la valeur des articles correspondants (la somme de toutes ces valeurs fournirait la ''valeur du stock'').

Remarques

Un nom de champ constitue, à lui tout seul, une expression dont la valeur n'est rien d'autre que celle du champ en question. Mais un nombre (ou constante) constitue également une expression ; toutefois dans ce cas, sa valeur ne dépend plus de l'enregistrement concerné de sorte que cette situation est peu usuelle puisque ne présentant guère d'intérêt. En voici un exemple ''d'école''.

```
. use repert
. list nom, 1, prenom
00001  Thomas       1 Michel
00002  Mitenne      1 Laurence
00003  Loiseau      1 Albert
00004  Meunier      1 Albert
00005  Taillefert   1 Claude
00006  Grillon      1 Jules
```

écran 6.8 : quand une expression se réduit à un simple nombre

3.2. Expressions de type caractère

Généralement, lorsque l'on parle d'expressions, on pense plutôt à des expressions numériques, par analogie avec les expressions algébriques que l'on rencontre en mathématiques. Mais, il existe également en Dbase des ''*expressions de type caractère*''. Ce sont des expressions qui fournissent un résultat de ce type. Ainsi un nom de champ tel que VILLE constitue un cas particulier d'expression de type caractère.

Constantes de type caractère

Un autre cas particulier réside dans les ''constantes de type caractère''. En voici un exemple :

```
. use repert
. list "bonjour", prenom
00001  bonjour Michel
00002  bonjour Laurence
00003  bonjour Albert
00004  bonjour Albert
00005  bonjour Claude
00006  bonjour Jules
```

écran 6.9 : expressions particulières de type caractère (nom de champ et constante)

L'écriture :

''bonjour''

représente simplement la *suite de caractères* (on dit aussi *''chaîne''*) : b, o, n, j, o, u et r. Les guillements sont indispensables à Dbase pour lui éviter de confondre le mot bonjour avec un nom de champ (ou, comme nous le verrons plus tard, avec un nom de variable).

Voici d'autres exemples d'expression de type caractère :

L'extraction de ''sous-chaînes'' : $

```
. use repert
. list $(nom,1,4)
00001  Thom
00002  Mite
00003  Lois
00004  Meun
00005  Tail
00006  Gril
```

écran 6.10 : la fonction $

L'expression :

$(NOM,1,4)

signifie : prendre, dans l'expression NOM (réduite ici à un champ), quatre caractères à partir de celui de rang 1.

La conversion en majuscules : !

```
. use repert
. list !(nom)
00001  THOMAS
00002  MITENNE
00003  LOISEAU
00004  MEUNIER
00005  TAILLEFERT
00006  GRILLON
```

écran 6.11 : conversion en majuscules

L'expression :

!(NOM)

signifie : la valeur du champ NOM, convertie en majuscules. (Bien entendu, le contenu du champ reste inchangé ; Dbase se contente d'effectuer la conversion avant d'en afficher le résultat).

La "concaténation" de deux chaînes : +

```
. use repert
. list nom+prenom
00001  Thomas     Michel
00002  Mitenne    Laurence
00003  Loiseau    Albert
00004  Meunier    Albert
00005  TaillefertClaude
00006  Grillon    Jules
. list nom, prenom
00001  Thomas      Michel
00002  Mitenne     Laurence
00003  Loiseau     Albert
00004  Meunier     Albert
00005  Taillefert Claude
00006  Grillon     Jules
```

écran 6.12 : concaténation de 2 chaînes

L'expression :

NOM + PRENOM

représente la chaîne obtenue en mettant "bout à bout" les valeurs des deux chaînes NOM et PRENOM. (Une telle opération entre chaînes porte le nom de *concaténation*). Notez que le résultat diffère (d'un espace !) de ce que produirait l'énumération des deux champs NOM

et PRENOM (Dbase affiche un espace entre les différentes expressions qu'on lui demande d'afficher).

La suppression des espaces de fin : TRIM

Dans l'expression NOM + PRENOM, les espaces de fin du champ NOM étaient conservés. La fonction TRIM, appliquée à une expression de type chaîne fournit une chaîne débarrassée des espaces de fin.

```
. use repert
. list trim(nom)+prenom
00001  ThomasMichel
00002  MitenneLaurence
00003  LoiseauAlbert
00004  MeunierAlbert
00005  TaillefertClaude
00006  GrillonJules
. list trim(nom), prenom
00001  Thomas Michel
00002  Mitenne Laurence
00003  Loiseau Albert
00004  Meunier Albert
00005  Taillefert Claude
00006  Grillon Jules
. list trim(nom)+" "+prenom
00001  Thomas Michel
00002  Mitenne Laurence
00003  Loiseau Albert
00004  Meunier Albert
00005  Taillefert Claude
00006  Grillon Jules
```

écran 6.13 : la fonction TRIM

3.3. D'une manière générale

Nous n'avons vu que quelques exemples d'expressions. A titre indicatif, disons qu'une expression en Dbase peut comporter les éléments suivants :

- **constantes** (numériques ou caractères)
- **noms de champs**
- **noms de "variables"** (nous ne les rencontrerons que plus tard)
- **opérations numériques :**
  + addition
  – soustraction
  * multiplication
  / division

  (comme en "algèbre", multiplication et division sont prioritaires sur addition et soustraction)

- **opérations entre chaînes** de caractères :
  + concaténation
  – concaténation avec élimination des espaces intermédiaires
- **parenthèses :** ( )
  (comme en algèbre elles servent à modifier les priorités des opérations)
- **fonctions**
  nous avons déjà rencontré $, ! et TRIM. Il en existe beaucoup d'autres ; nous en rencontrerons certaines par la suite.

▷ **Entraînez-vous**

- Sur le fichier REPERT, essayez ces commandes :

```
LIST  TAILLE + 1
LIST  PRENOM, "mesure", TAILLE, "m"
LIST  TRIM(PRENOM), "mesure", TAILLE, "m"
LIST  PRENOM, "mesure", TAILLE + 0.2, "m"
LIST  $(NOM,1,3), $(VILLE,1,4)
LIST  $(NOM,1,3) + $(VILLE1,4)
LIST  $(NOM,4,9)
LIST  !(PRENOM)
LIST  !(VILLE)
LIST  !($(NOM,1,4))
```

- Que pensez-vous de ces commandes ? Vérifiez vos prévisions.

```
LIST  CODEP + 1
LIST  NOM + PRENOM
LIST  CODEP + "1"
LIST  $(NOM,6,10)
LIST  $(NOM,12,5)
LIST  BONJOUR, PRENOM
LIST  !(CODEP)
LIST  !(TAILLE)
LIST  !("bonjour" + NOM)
```

- Sur le fichier STOCK, essayez ces commandes :

```
LIST  "le produit", PRODUIT, "coute", PRIX, "F"
LIST  "il reste", QTE, "unités de", PRODUIT
LIST  "il reste", QTE-10, "unités de", PRODUIT
LIST  QTE, "articles à", PRIX, "ça fait", PRIX*QTE, "francs"
```

4. POUR EXPÉRIMENTER VOS PROPRES EXPRESSIONS

Si vous essayez de réaliser une expression quelque peu compliquée, il pourra vous arriver d'avoir quelques doutes sur ce qu'elle produit.

Il est toujours possible de placer votre expression dans une commande DISPLAY pour voir quelle sera sa valeur lorsqu'elle est appliquée à l'enregistrement courant. (Vous obtenez également le numéro de l'enregistrement en question). Une autre façon de procéder consiste à utiliser la commande :

?

Elle demande à Dbase d'afficher la valeur de l'expression (ou des expressions) qui la suit.

VII Pour imposer vos conditions

Jusqu'ici, notre façon d'examiner un fichier a été basée sur l'emploi du *''numéro d'enregistrement''*. Par exemple, nous consultions le *cinquième* enregistrement ou encore nous en affichions trois consécutifs à partir du quatrième, etc... Nous pouvons dire que notre recherche était tributaire de la manière dont notre fichier était ''physiquement'' organisé. Ou encore, que nous étions amenés à retrouver un enregistrement par sa position et non par son contenu.

Nous allons maintenant apprendre à retrouver des enregistrements à partir de conditions portant sur leur contenu. Par exemple, nous pourrons rechercher dans REPERT tous les habitants de PARIS ; ou encore, nous pourrons retrouver toutes les personnes mesurant plus de 1 m 72. Mieux encore, nous pourrons rechercher tous les ''Albert'', habitant PARIS et mesurant entre 1 m 68 et 1 m 75 !

1. POUR METTRE DES CONDITIONS : FOR(*)

Commençons par un exemple simple d'une condition placée dans une commande LIST

(*) En anglais : pour

```
. use stock
. list for qte<20
00002  Four à micro-ondes    E 248   1875.25     7
00004  Four à raclette 6 P   A 427    133.72    15
00006  Table de cuisson      E 647   2450.00     6
```

écran 7.1 : liste conditionnelle (1)

La commande :

LIST FOR QTE < 20

signifie : lister les enregistrements pour lesquels la condition suivante est vraie :

QTE < 20

Cette condition exprime simplement que la valeur du champ QTE doit être *inférieure* à 20.

Voici deux autres exemples :

```
. use repert
. list nom, prenom for ville="PARIS"
00004  Meunier     Albert
00006  Grillon     Jules
. use stock
. list produit, prix for qté*prix < 5000
00001  Cafetière 12 T          137.47
00003  Grille pain             158.75
00004  Four à raclette 6 P     133.72
```

écran 7.2 : liste conditionnelle (2)

La première commande :

LIST NOM, PRENOM FOR VILLE = ''PARIS''

utilise la condition :

VILLE = ''PARIS''

Celle-ci est vérifiée lorsque le contenu du champ VILLE est égal à la constante caractère ''PARIS''. Notez à nouveau les guillemets. En leur absence, une condition telle que :

VILLE = PARIS

amènerait Dbase à considérer PARIS comme un nom de champ (ou de variable !).

Le second exemple est basé sur la condition :

QTE* PRIX < 5000

Elle nous permet de sélectionner les articles dont la valeur en stock est inférieure à 5000 F.

En fait, des ''conditions'' telles que celles que nous venons d'évoquer peuvent être utilisées dans d'autres commandes que LIST (ou DISPLAY). Dans tous les cas, elles sont constituées suivant les mêmes principes que nous allons vous exposer progressivement ici.

2. LES CONDITIONS LES PLUS USITÉES : LES COMPARAISONS

Les trois conditions que nous venons d'évoquer :

QTE < 20

VILLE = ''PARIS''

QTE * PRIX < 5000

rentrent dans cette catégorie. D'une manière générale, on peut comparer entre elles n'importe quelles expressions pour peu qu'elles soient de même type. Cela se fait à l'aide de ce que l'on nomme un ''*opérateur de comparaison*'' (< dans le premier cas, = dans le second, > dans le troisième).

2.1. Les opérateurs de comparaison

Ils sont au nombre de 6. Ce sont <, >, =, < =, > = et < > (différent de). Ils sont employés aussi bien pour les expressions numériques que pour les expressions de type caractère.

Leur signification est assez facile à deviner dans le premier cas. Par contre, les choses sont un peu moins simples dans le second. Que peut en effet signifier < dans le cas d'expressions de type caractère ? On peut songer, à juste titre, à un ''*ordre alphabétique*''. Mais encore faut-il pouvoir tenir compte des majuscules, des minuscules, des chiffres et... des autres caractères (+ – , ; <, etc...)

En fait, il faut savoir que les caractères sont ordonnés suivant la valeur de leur ''code''. A chaque caractère est associé un nombre ; c'est ce

nombre, codé en "binaire" qui sert à représenter le caractère en mémoire. Dans le cas de Dbase, le code utilisé est l'ASCII (abréviation de American Standard Code for Information Interchange : code américain standard pour l'échange d'information). Dans ce code, les *chiffres* apparaissent *avant* les *majuscules* qui apparaissent elles-mêmes *avant* les *minuscules*.

Ainsi donc, par exemple, < signifiera "situé avant" au sens de l'ordre des caractères ainsi défini.

Voici une récapitulation de la signification des opérateurs de comparaison, suivant qu'ils portent sur des expressions numériques ou de type caractère :

Opérateur de comparaison	Signification pour des expressions numériques	Signification pour des expressions de type caractère
=	égal à	(*) égal à
<	inférieur à	placé avant
>	supérieur à	placé après
< =	inférieur ou égal à	(*) placé avant ou égal
> =	supérieur ou égal à	(*) placé après ou égal
< >	différent de	(*) différent de

tableau 7.1 : les opérateurs de comparaison

2.2. Les expressions à comparer

Dans nos exemples d'introduction, l'une de nos deux expressions était une constante (5, "PARIS" ou 5000). Bien entendu, il ne s'agit aucunement d'une obligation, même si cette situation est relativement fréquente.

Voici quelques autres exemples :

```
. use repert
. list off nom, prenom, ville for nom<ville
Loiseau     Albert    MORREZ
Meunier     Albert    PARIS
Grillon     Jules     PARIS
. list off for !(prenom)="ALBERT"
Loiseau     Albert    39400 MORREZ    84 61 28 42   1.71
Meunier     Albert    75008 PARIS     49 33 27 65   1.83
```

écran 7.3 : listes conditionnelles (3)

(*) L'égalité pouvant être "partielle" suivant la valeur de l'indicateur "EXACT" (voir paragraphe 3)

Le premier n'est qu'un exemple ''d'école'' dans la mesure où il a peu de chance d'être utilisé en pratique. Il fournit toutes les personnes pour lesquelles le nom précède le nom de la ville (au sens du code ASCII).

Le second, par contre, correspond à un usage réel : il permet d'obtenir toutes les personnes de prénom Albert, sans tenir compte des majuscules ou des minuscules.

▷ **Entraînez-vous**

(Les points comportant un numéro sont ''corrigés'' en fin de volume)

- (1) Listez tous les enregistrements du fichier STOCK dont le prix est d'au moins 200 F.
- (2) Listez le nom et le nombre des produits du fichier STOCK pour lesquels il reste moins de 5 unités.
- (3) Dans les 5 premiers enregistrements de REPERT, listez toutes les personnes dont le nom apparaît avant ''MITENNE'' dans l'ordre alphabétique (en ne tenant pas compte des majuscules et des minuscules).
- (4) Listez toutes les personnes dont le prénom commence par un A.
- (5) Listez en les écrivant en majuscules, tous les noms des personnes mesurant plus de 1,68 m.
- (6) Affichez les quatre premières lettres du nom et les trois premières du prénom pour toutes les personnes dont la première lettre de la ville est un P.
- Que pensez-vous des commandes suivantes (appliquées au fichier REPERT) ? Vérifiez vos suppositions
  LIST FOR CODEPOST < 45000
  LIST ! (NOM) FOR $(PRENOM,2,1)="E"

3. L'EXACTITUDE DES COMPARAISONS D'ÉGALITÉ POUR LES CARACTÈRES : SET EXACT

Voyez ces deux commandes de liste.

```
. use repert
. list for nom="Mitenne"
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
. list for nom="Mit"
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
```

écran 7.4 : égalité partielle !

Apparemment, Dbase considère que le contenu du champ NOM est égal à la chaîne ''Mit'' lorsqu'il contient ''Mitenne''. Effectivement, l'égalité de deux chaînes est jugée vraie dès lors que la seconde chaîne correspond exactement au début de la première.

Cette situation vous gêne et vous souhaitez que l'égalité ne soit considérée comme vraie que lorsque les deux chaînes sont identiques ! Qu'à cela ne tienne, il suffit de le spécifier à Dbase par une commande :

SET EXACT ON

Si, ultérieurement, vous souhaitez revenir à des ''égalités partielles'', il vous suffira d'exécuter la commande contraire :

SET EXACT OFF

Voyez ce que devient notre précédent exemple :

```
. use repert
. set exact on
. list for nom="Mit"
. set exact off
. list for nom="Mit"
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
```

écran 7.5 : égalité exacte ou partielle

Remarque :

Il existe de nombreux ''indicateurs'' tels que EXACT qui peuvent prendre l'une des valeurs ON (''vrai'' ou ''activé'') ou OFF (''faux'' ou ''désactivé''). Nous en rencontrerons d'autres par la suite. Ils seront toujours mis en place par une commande SET. Pour connaître ''l'état'' de ces indicateurs, vous pouvez employer la commande :

DISPLAY STATUS

4. LA CONDITION D'APPARTENANCE : $

```
. use repert
. list off for "il" $ nom
Taillefert Claude    45510 SULLY     77 89 63 10   1.78
Grillon    Jules     75006 PARIS     42 48 15 30   1.68
. list off for "u" $ prenom
Mitenne    Laurence 83600 FREJUS    89 55 66 89   1.58
Taillefert Claude    45510 SULLY     77 89 63 10   1.78
Grillon    Jules     75006 PARIS     42 48 15 30   1.68
```

écran 7.6 : les noms contenant ''il'' et les prénoms contenant ''u''

La première condition :

''il'' $ NOM

est vraie lorsque la chaîne **il** est contenue dans le champ NOM. La seconde condition permet de sélectionner toutes les personnes dont le prénom comporte la lettre u (minuscule).

Remarque :

Ne confondez pas la condition d'appartenance avec l'extraction de sous-chaîne (bien que la ''notation'' en soit la même). D'autre part, ne confondez pas cette *condition d'appartenance* qui correspond à la ''présence'' de la première chaîne dans la seconde (et ceci quelle que soit sa place) avec *l'égalité partielle* qui correspond au cas où la seconde chaîne est le début de la première.

▷ **Entraînez-vous**

(Les points comportant un numéro sont corrigés en fin de volume)

- (7) Listez toutes les personnes du fichier REPERT comportant la lettre A dans leur nom (en majuscule ou en minuscule)
- (8) Lister toutes les personnes comportant le groupe 86 dans leur numéro de téléphone.

5. POUR RETROUVER UN PAR UN LES ENREGISTREMENTS VÉRIFIANT UNE CONDITION : LOCATE FOR ET CONTINUE

La commande :

LOCATE FOR(*)

(*) Locate, en anglais signifie : situer.

permet de placer le pointeur sur le *premier enregistrement* du fichier vérifiant une condition donnée et d'en afficher son numéro. Si on le désire, la commande :

CONTINUE

permet de rechercher le prochain enregistrement satisfaisant à la même condition (celle qui a été précisée dans le précédent LOCATE). Cette instruction CONTINUE peut être utilisée autant de fois que vous le souhaitez.

```
. use repert
. locate for ville="PARIS"
*** ENREGISTREMENT : 00004
. display
00004  Meunier     Albert    75008 PARIS     49 33 27 65   1.83
. continue
*** ENREGISTREMENT : 00006
. display
00006  Grillon     Jules     75006 PARIS     42 48 15 30   1.68
. continue
*** FIN DE FICHIER ***
```

écran 7.7 : localisation conditionnelle d'enregistrements

Notez bien que ni LOCATE, ni CONTINUE n'affichent le contenu de l'enregistrement trouvé. Ces deux commandes se révèlent surtout utiles lorsqu'elles sont incorporées dans un *programme*.

Voyez comme Dbase signale qu'il a atteint la ''fin du fichier'' sans trouver d'enregistrement vérifiant la condition.

6. LES CONDITIONS MULTIPLES

Jusqu'ici, nos conditions étaient soit une comparaison, soit une condition d'appartenance. En fait, Dbase vous permet d'écrire des combinaisons de telles conditions. En voici un premier exemple :

```
. use repert
. list for prenom="Albert" .and. ville="PARIS"
00004  Meunier     Albert    75008 PARIS     49 33 27 65   1.83
```

écran 7.8. : l'opérateur logique .AND.

La condition :

PRENOM = "Albert" .AND. VILLE = "PARIS"

est vraie lorsque les deux conditions :

PRENOM = "Albert"
VILLE = "PARIS"

sont vraies.

.AND. s'appelle un *"opérateur logique"*. Il existe deux autres opérateurs logiques(*) :

.OR.
.NOT.

(N'oubliez pas les points qui "encadrent" ces mots ! Ils permettent à Dbase de ne pas les confondre avec d'éventuels noms de champ ou de variable. Ne laissez pas d'espace entre les points et le mot !)

```
. use repert
. list for prenom="Albert" .or. ville="PARIS"
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
```

écran 7.9 : l'opérateur logique .OR.

La condition :

PRÉNOM = "Albert" .OR. VILLE = "PARIS"

est vraie lorsque *l'une au moins* des deux conditions :

PRENOM = "Albert"
VILLE = "PARIS"

est vraie (les deux pouvant être vraies simultanément)

```
. use repert
. list for .not. ("a" $ nom)
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
```

écran 7.10 : l'opérateur logique .NOT.

(*) En anglais, AND signifie ET, OR signifie OU et NOT signifie NON (ou PAS).

La condition :

.NOT. ("a" $ NOM)

est vraie lorsque la condition

"a" $ NOM

est fausse. Nous obtenons ainsi tous les enregistrements pour lesquels le nom ne comporte pas la lettre a.

▷ **Entraînez-vous**

(Les points comportant un numéro sont corrigés en fin de volume)

- (9) A partir du fichier REPERT :

– listez toutes les personnes habitant PARIS, dont le nom commence par la lettre T
– listez toutes les personnes se prénommant Albert ou Claude.

- (10) A partir du fichier STOCK :

– listez tous les articles de catégorie A coûtant moins de 150 F.
– listez tous les articles de catégorie E dont la valeur en stock dépasse 14 000 F.

7. POUR COMPTER LE NOMBRE D'ENREGISTREMENTS VÉRIFIANT UNE CONDITION : COUNT.

Voyez cet exemple d'utilisation de la commande :

COUNT(*)

```
. use repert
. count for prenom="Albert"
** COMPTEUR = 00002
```

écran 7.11 : pour compter le nombre de personnes se prénommant Albert.

Remarques

1) Dbase accepte COUNT (tout court). Cette commande ne fait alors rien d'autre que de compter le nombre total d'enregistrements !

(*) En anglais : compter.

2) Nous n'avons pas pris soin d'activer l'indicateur EXACT. De sorte que nous pourrions obtenir également tous les « Abertin » ou toutes les ''Albertine''...

8. POUR FAIRE DES SOMMES : SUM

La commande :

SUM(*)

permet de totaliser les valeurs d'expressions numériques. Comme COUNT, elle peut soit porter sur tout le fichier, soit être assortie d'une condition.

8.1 Pour faire des sommes sur tout le fichier

```
. use stock
. sum qte
  104
```

écran 7.12 : calcul du nombre total d'articles en stock

La commande :

SUM QTE

signifie : effectuer la somme des valeurs du champ QTE de tous les enregistrements du fichier courant.

```
. use stock
. sum qte*prix
 45217.09
```

écran 7.13 : calcul de la valeur du stock

Cette fois, ce sont les valeurs de l'expression

QTE * PRIX

qui sont totalisées. Nous obtenons ainsi la valeur de l'ensemble du stock.

(*) En anglais : faire la somme.

8.2 Pour faire des sommes conditionnelles

Il suffit d'ajouter une condition dans la commande SUM à l'aide de FOR. En voici deux exemples :

```
. use stock
. sum qte for cat="A"
  70
```

écran 7.14 : calcul du nombre d'articles de catégorie A

```
. use stock
. sum qte*prix for cat="E"
 27826.75
```

écran 7.15 : calcul de la valeur du stock des articles de catégorie E

VIII La mise à jour "automatique"

Nous avons déjà étudié ce que nous appelions la mise à jour "à vue". Nous y avons vu comment mettre à jour un fichier, en travaillant sur un seul enregistrement à la fois d'une manière qu'on pourrait qualifier de visuelle et de manuelle. Nous allons maintenant découvrir comment Dbase nous permet d'effectuer les choses d'une manière relativement "automatique".

En particulier, nous allons apprendre comment, à l'aide d'une seule commande :

— effacer plusieurs enregistrements
— effectuer des modifications sur plusieurs enregistrements.

Auparavant, et afin de pouvoir manipuler en toute quiétude sur nos fichiers, nous allons apprendre à en faire une "copie de sécurité" à l'aide de la commande COPY TO...

1. POUR RECOPIER DES FICHIERS : COPY TO

Comme nous le verrons plus tard, la commande :

COPY TO(*)

(*) En anglais : recopier dans.

offre des possibilités très variées. Pour l'instant, nous allons simplement l'employer pour effectuer des copies de nos deux fichiers REPERT et STOCK.

```
. use repert
. copy to repert1
00006 ENREGISTREMENT(S) TRANSFERE(S)
. list files

NOMS FICHIERS     ENREG.     MIS A JOUR
REPERT    DBF     00006      08/06/86
STOCK     DBF     00006      08/06/86
REPERT1   DBF     00006      08/06/86


. use repert1
. list
00001  Thomas      Michel    58000 NEVERS    86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS    89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS     49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY     77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS     42 48 15 30  1.68
```

écran 8.1 : création d'une copie de sécurité du fichier REPERT

Ici, nous avons recopié le fichier REPERT dans un nouveau fichier que nous avons choisi de nommer REPERT1. Notez bien que, dans la commande :

COPY TO REPERT1

nous ne nommons que le *fichier destination* (c'est-à-dire celui qui va recevoir la copie). Le *fichier de départ* est le fichier associé à la zone de travail, c'est-à-dire celui qui a été préalablement nommé dans une commande USE.

▷ **Entraînez-vous**

- Procédez de façon analogue pour recopier le fichier STOCK dans STOCK1.

2. *POUR EFFACER PLUSIEURS ENREGISTREMENTS AVEC DELETE*

Nous avons déjà vu comment marquer pour effacement un enregistrement. Ainsi :

DELETE marque l'enregistrement "courant"
DELETE RECORD 3 marque l'enregistrement numéro 3.

En fait, au même titre que de nombreuses commandes de Dbase, DELETE peut être assortie :

— d'une indication de portée ; par exemple :

ALL NEXT 5

(RECORD 3 étant d'ailleurs également une indication de portée particulière)

— d'une condition FOR

— éventuellement simultanément d'une portée et d'une condition.

3. *LA COMMANDE DELETE AVEC INDICATION DE "PORTÉE"*

Voici un premier exemple de marquage de 3 enregistrements consécutifs :

```
. use repert
. list
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. goto 2
. delete next 3
00003 ENREGISTREMENT(S) EFFACE(S)
. list
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002 *Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003 *Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00004 *Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. recall all
00003 ENREGISTREMENT(S) RESTITUE(S)
```

écran 8.2 : marquage de 3 enregistrements consécutifs

Et un autre, où l'on marque tous les enregistrements :

```
. use repert
. delete all
00006 ENREGISTREMENT(S) EFFACE(S)
. list
00001 *Thomas      Michel    58000 NEVERS    86 48 23 37  1.75
00002 *Mitenne     Laurence  83600 FREJUS    89 55 66 89  1.58
00003 *Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
00004 *Meunier     Albert    75008 PARIS     49 33 27 65  1.83
00005 *Taillefert  Claude    45510 SULLY     77 89 63 10  1.78
00006 *Grillon     Jules     75006 PARIS     42 48 15 30  1.68
. recall all
00006 ENREGISTREMENT(S) RESTITUE(S)
```

écran 8.3 : marquage de tous les enregistrements d'un fichier

Ce deuxième exemple doit plutôt être considéré comme un "cas d'école", dans la mesure où il a peu de chances d'être utilisé dans la pratique.

4. EFFACEMENT CONDITIONNEL

Voici, par exemple, comment marquer tous les enregistrements du fichier REPERT, correspondants à des personnes mesurant moins de 1 m 73.

```
. use repert
. delete for taille < 1.73
00003 ENREGISTREMENT(S) EFFACE(S)
. list
00001  Thomas      Michel    58000 NEVERS    86 48 23 37  1.75
00002 *Mitenne     Laurence  83600 FREJUS    89 55 66 89  1.58
00003 *Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS     49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY     77 89 63 10  1.78
00006 *Grillon     Jules     75006 PARIS     42 48 15 30  1.68
. recall all
00003 ENREGISTREMENT(S) RESTITUE(S)
```

écran 8.4 : effacement conditionnel

Remarques

1) Vous pouvez, si vous le souhaitez, conjuguer portée et condition, comme dans cet exemple :

DELETE NEXT 4 FOR TAILLE < 1.70

2) La commande RECALL peut, de manière similaire, être assortie d'une portée et d'une condition.

3) Lorsqu'une *condition* est présente, en l'absence de portée, elle *concerne par défaut tout le fichier*. (C'est donc comme si vous indiquiez ALL)

▷ **Entraînez-vous**

(les points comportant un numéro sont corrigés en fin de volume)

- (1) Marquez pour effacement tous les enregistrements de REPERT pour lesquels le code postal est inférieur à 46000.
- Supprimez ces marques.
- (2) Marquez pour effacement tous les enregistrements de REPERT pour lesquels le nom comporte la lettre L (majuscule ou minuscule).
- Supprimez ces marques.

5. POUR EFFECTUER DES REMPLACEMENTS AUTOMATIQUES : REPLACE

La commande DELETE vous était déjà connue en partie. Par contre nous n'avons pas encore rencontré la ocmmande :

REPLACE(*)

Son principal intérêt sera de permettre d'effectuer des modifications sur plusieurs enregistrements. Néanmoins, pour vous familiariser avec cette commande, nous allons commencer par vous la présenter *sans portée ni condition,* de sorte qu'elle portera alors uniquement sur l'enregistrement courant (Notez, d'ailleurs, qu'en *programmation,* il sera courant d'utiliser cette commande dans de telles conditions.)

(*) En anglais : remplacer.

```
. use repert
. goto 3
. display
00003  Loiseau     Albert    39400 MORREZ    84 61 28 42  1.71
. replace nom with "Loiselet"
00001 REMPLACEMENT(S)
. list
00001  Thomas      Michel    58000 NEVERS    86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS    89 55 66 89  1.58
00003  Loiselet    Albert    39400 MORREZ    84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS     49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY     77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS     42 48 15 30  1.68
```

écran 8.5 : remplacement de la valeur
d'un champ de l'enregistrement courant (1)

La commande :

REPLACE NOM WITH ''Loiselet''

signifie : remplacer la valeur actuelle du champ NOM par la valeur indiquée à la suite du mot WITH, ici la constante chaîne : Loiselet.

Certes, la quantité ''remplaçante'' était ici une constante, mais il peut s'agir d'une expression quelconque. Voici d'autres exemples :

```
. use repert
. goto 2
. display
00002  Mitenne     Laurence  83600 FREJUS    89 55 66 89  1.58
. replace nom with prenom
00001 REMPLACEMENT(S)
. display
00002  Laurence    Laurence  83600 FREJUS    89 55 66 89  1.58
```

écran 8.6 : remplacement du nom par le prénom
dans l'enregistrement courant

```
. use repert
. goto 3
. display
00003  Loiselet    Albert    39400 MORREZ    84 61 28 42  1.71
. replace nom with !(nom)
00001 REMPLACEMENT(S)
. display
00003  LOISELET    Albert    39400 MORREZ    84 61 28 42  1.71
```

écran 8.7 : passage en majuscule du nom de l'enregistrement courant

```
. use stock
. goto 4
. display
00004  Four à raclette 6 P  A 427    133.72     15
. replace qte with qte+6
00001 REMPLACEMENT(S)
. display
00004  Four à raclette 6 P  A 427    133.72     21
```

écran 8.8 : augmentation de 6 du nombre d'articles de l'enregistrement courant

▷ **Entraînez-vous**

(les points comportant un numéro sont corrigés en fin de volume)

Toutes les manipulations proposées portent sur le fichier REPERT

- (3) Remplacez par Michel, le prénom de l'enregistrement numéro 4.
- (4) Remplacez par 58100 le code postal de l'enregistrement numéro 1
- (5) Remplacez par 1.76 la taille de l'enregistrement numéro 1
- (6) Transformez en majuscules le prénom de l'enregistrement numéro 2.
- Voyez ce qui se passe si vous cherchez à exécuter chacune de ces commandes (sur un enregistrement de votre choix) :

  REPLACE NOM WITH "Nombienlong"
  REPLACE CODEPOS WITH 45000
  REPLACE TAILLE WITH "GRAND"
  REPLACE PRENOM WITH Leon

- Voyez comme Dbase accepte une commande telle que

  REPLACE NOM WITH CODEPOS

alors qu'il refuse

  REPLACE PRENOM WITH TAILLE

6. LA COMMANDE REPLACE AVEC INDICATION DE "PORTÉE"

Nous vous en proposons deux exemples avec ALL

a) **Pour transformer en majuscules un champ caractère.**

```
. use repert
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Laurence    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  LOISELET    Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
. replace all nom with !(nom)
00006 REMPLACEMENT(S)
. list
00001  THOMAS      Michel    58000 NEVERS   86 48 23 37  1.75
00002  LAURENCE    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  LOISELET    Albert    39400 MORREZ   84 61 28 42  1.71
00004  MEUNIER     Albert    75008 PARIS    49 33 27 65  1.83
00005  TAILLEFERT  Claude    45510 SULLY    77 89 63 10  1.78
00006  GRILLON     Jules     75006 PARIS    42 48 15 30  1.68
```

écran 8.9 : pour transformer tous les noms en majuscules

b) **Pour mettre une majuscule en début de nom de produit**

Pour l'instant, nos noms sont écrits ainsi. Pour que l'exemple soit "probant", nous vous suggérons de modifier par EDIT tout ou partie des noms pour qu'il n'en soit plus ainsi, avant d'expérimenter l'exemple suivant :

```
. use stock
. list
00001  cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  grille pain           A 521    158.75     23
00004  four à raclette 6 P   A 427    133.72     21
00005  Friteuse 1 kg         B 433    349.25     21
00006  table de cuisson      E 647   2450.00      6
. replace all produit with $(produit,1,1) + $(produit,2,19)
00006 REMPLACEMENT(S)
. list
00001  cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  grille pain           A 521    158.75     23
00004  four à raclette 6 P   A 427    133.72     21
00005  Friteuse 1 kg         B 433    349.25     21
00006  table de cuisson      E 647   2450.00      6
```

écran 8.10 : pour mettre systématiquement une majuscule au début de chaque nom de produit.

7. REMPLACEMENTS CONDITIONNELS

Ils seront réalisés avec une condition FOR incorporée dans une commande REPLACE.

Voici un premier exemple ; nous supposons que, par erreur, la taille des « Parisiens » a été saisie avec une erreur de 2 cm (en moins). Nous pouvons ''rectifier'' tout cela :

```
. use repert
. list
00001  THOMAS      Michel    58000 NEVERS   86 48 23 37  1.75
00002  LAURENCE    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  LOISELET    Albert    39400 MORREZ   84 61 28 42  1.71
00004  MEUNIER     Albert    75008 PARIS    49 33 27 65  1.85
00005  TAILLEFERT  Claude    45510 SULLY    77 89 63 10  1.78
00006  GRILLON     Jules     75006 PARIS    42 48 15 30  1.70
. replace taille with taille+0.02 for ville = "PARIS"
00002 REMPLACEMENT(S)
. list
00001  THOMAS      Michel    58000 NEVERS   86 48 23 37  1.75
00002  LAURENCE    Laurence  83600 FREJUS   89 55 66 89  1.58
00003  LOISELET    Albert    39400 MORREZ   84 61 28 42  1.71
00004  MEUNIER     Albert    75008 PARIS    49 33 27 65  1.87
00005  TAILLEFERT  Claude    45510 SULLY    77 89 63 10  1.78
00006  GRILLON     Jules     75006 PARIS    42 48 15 30  1.72
```

écran 8.11 : pour augmenter de 2 cm la taille des ''parisiens''

La commande :

REPLACE TAILLE WITH TAILLE + 0,02 FOR !(VILLE) = ''PARIS''

signifie : remplacer la valeur du champ TAILLE par l'expression :

TAILLE + 0.02

et ceci pour tous les enregistrements qui vérifient la condition

!(VILLE) = ''PARIS''

(Notez que nous avons comparé à PARIS, non pas la valeur du champ VILLE, mais cette valeur convertie en majuscules ; Ceci nous permet de sélectionner tous les enregistrements où le nom de la capitale est écrit PARIS, Paris ou même PaRiS !)

Voici un autre exemple, dans lequel nous augmentons de 10 % le prix de tous les articles de catégorie A du fichier STOCK.

```
. use stock
. list
00001  cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  grille pain           A 521    158.75     23
00004  four à raclette 6 P   A 427    133.72     21
00005  Friteuse 1 kg .       B 433    349.25     21
00006  table de cuisson      E 647   2450.00      6
. replace prix with prix*1.1 for cat ="A"
00003 REMPLACEMENT(S)
. list
00001  cafetière 12 T        A 432    151.21     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  grille pain           A 521    174.62     23
00004  four à raclette 6 P   A 427    147.09     21
00005  Friteuse 1 kg         B 433    349.25     21
00006  table de cuisson      E 647   2450.00      6
```

écran 8.12 : pour augmenter de 10 % les prix des articles de catégorie A

REMARQUE

N'oubliez pas que, en l'absence d'indicateur de portée, la condition FOR concerne, par défaut, *tout le fichier*.

▷ Entraînez-vous

(Les points comportant un numéro sont corrigés en fin de volume).

Toutes les manipulations portent sur le fichier STOCK que vous aurez remis dans l'état "initial" en y recopiant le fichier STOCK1 créé au début de ce chapitre. Il doit donc se présenter ainsi :

```
. list
00001  Cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
00005  Friteuse 1 kg         B 433    349.25     21
00006  Table de cuisson      E 647   2450.00      6
```

- (7) Augmentez tous les prix de 8 %.
- (8) Revenez à l'état initial par une seule commande REPLACE (vérifiez !)
- (9) Diminuez de 20 % le prix des articles dont le nombre est inférieur à 20.
- (10) Diminuez de 5 le nombre de tous les articles de catégorie A.
- (11) Augmentez de 12 % le prix de tous les articles dont le nombre est compris entre 15 et 30 (pensez aux conditions multiples !)

IX Le tri : une (certaine) façon de mettre de l'ordre

Lorsque vous demandez une liste d'un fichier, les enregistrements apparaissent dans l'ordre où vous les avez créés. De même, lorsque vous effectuez une recherche (LOCATE FOR..., DISPLAY FOR...), Dbase explore les enregistrements suivant cet ordre qui vous paraît probablement naturel.

Mais, il est certainement des circonstances où cet ordre ne vous satisfait plus totalement. Par exemple, vous pouvez souhaiter obtenir une liste du fichier REPERT ordonnée suivant l'ordre alphabétique des noms.

Pour obtenir un tel résultat de Dbase il existe deux méthodes relativement différentes : le *tri* et *l'indexation.*

Le *tri* consiste à fabriquer, à partir d'un fichier existant, un nouveau fichier dans lequel les enregistrements sont rangés dans l'ordre voulu. Cette opération crée un second fichier de même taille que le premier. Vous pouvez par la suite faire l'usage que vous voulez de chacun de ces deux fichiers. Vous pouvez éventuellement supprimer le premier.

L'*indexation,* quant à elle, consiste à fabriquer une table de correspondance (nommée index) entre l'information intéresante de l'enregistrement et le numéro d'enregistrement correspondant. Elle ne fabrique donc pas de nouveau fichier, exception faite de la table d'index.

Ce chapitre est consacré au tri. L'indexation sera étudiée dans les chapitres suivants.

1. POUR TRIER SUR UN CHAMP DE TYPE CARACTÈRE

Considérons le fichier REPERT tel qu'il apparaît en page 66 (Vous pouvez toujours lui redonner cet état en y recopiant REPERT1). Voyons comment créer un nouveau fichier *trié sur le champ NOM,* à l'aide de la commande :

SORT(*)

```
. use repert
. sort on nom to nrepert
TRI TERMINE
. list files

NOMS FICHIERS     ENREG.     MIS A JOUR
REPERT    DBF     00006      00/00/00
STOCK     DBF     00006      09/06/86
REPERT1   DBF     00006      08/06/86
STOCK1    DBF     00006      09/06/86
NREPERT   DBF     00006      10/06/86
```

écran 9.1 : tri du fichier REPERT sur le nom

La commande :

SORT ON NOM TO NREPERT

signifie : Trier (SORT) le fichier sur (ON) le champ NOM en plaçant le fichier ainsi trié dans NREPERT.

Voyez comme la commande LIST FILES nous confirme que Dbase a effectivement créé un nouveau fichier nommé NREPERT, sans pour autant supprimer REPERT.

Comparons ces deux fichiers :

(*) En anglais : trier.

```
. use repert
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
. use nrepert
. list
00001  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
00002  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00003  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00004  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
```

écran 9.2 : comparaison entre fichier initial et fichier trié

2. QUEL ORDRE POUR LES CARACTÈRES ?

Nous avons déjà signalé que les caractères étaient rangés suivant l'ordre de leur code. Bien entendu, tant que vous n'utilisez, par exemple, que des lettres majuscules, tout va très bien. Vous retrouvez en effet le bon vieil ordre alphabétique si cher aux dictionnaires.

Par contre, les choses se gâtent quelque peu dès que vous mélangez les genres :

— lettres et chiffres
— majuscules et minuscules
— lettres et caractères accentués
— etc...

D'une part, comme nous l'avons déjà vu, les chiffres apparaissent avant les majuscules qui, elles mêmes, apparaissent avant les minuscules. Ce phénomène conduit déjà à des situations peu naturelles.

Mais, d'autre part, les choses sont encore moins claires si l'on emploie les caractères dits ''nationaux'', comme à, é,è,ç, etc... En effet, certains apparaissent avant les chiffres, d'autres après les minuscules.

D'autre part, pour lever un doute, vous pouvez toujours demander à Dbase de vous afficher le code d'un caractère donné, grâce à la fonction RANK. Plus précisément, l'expression :

RANK (''é'')

désigne le ''rang'' (ou code) de la constante chaîne ''é''.

Nous vous rappelons que nous pouvons afficher la valeur d'une expression à l'aide de la commande ? (nous obtiendrions le même résultat avec DISPLAY OFF mais à condition qu'un fichier soit en cours d'utilisation...)

```
. ? rank("a")
  97
. ? rank("é")
 123
. ? rank("ç")
  92
. ? rank("*")
  42
. ? rank("=")
  61
. ? rank("A")
  65
```

écran 9.3 : pour obtenir le code d'un caractère : la fonction RANK

Réciproquement, bien que cela présente moins d'intérêt, vous pouvez également demander à Dbase de vous afficher un caractère de code donné, grâce à la fonction CHR.

```
. ? chr(97)
a
. ? chr(65)
A
. ? chr(98)
b
. ? chr(33)
!
```

écran 9.4 : pour obtenir un caractère de code donné : la fonction CHR

3. QUAND DES CHIFFRES SE TROUVENT DANS UN CHAMP CARACTÈRE

Si vous cherchez à classer les deux chaînes PARIS2 et PARIS12, vous constatez qu'elles apparaissent dans l'ordre suivant :

PARIS12
PARIS2

En effet, les cinq premiers caractères de ces deux chaînes sont identiques. Le sixième, par contre, est différent. C'est donc lui qui va permettre d'effectuer le classement, et ceci sans tenir compte des éventuels caractères suivants. Ce phénomène, peut-être inattendu lorsqu'il s'agit de chiffres, est cependant le même que celui qui permet de classer le mot RANGEMENT avant le mot RANGER :

RANGEMENT
RANGER

Le phénomène peut devenir franchement gênant lorsque vous placez dans des *champs de type caractères,* des codes formés essentiellement de *chiffres* dont le nombre peut varier. Par exemple, supposons que dans un champ caractère de longueur 4, vous ayez été amenés à entrer 423 d'une part et 1285 d'autre part. Ces informations (considérées comme des chaînes de caractères) vont se retrouver "cadrées" à gauche du champ, comme l'illustre ce shéma :

4	2	3	

1	2	8	5

Dans ces conditions, une comparaison de ces deux informations conduira à classer 1285 *avant* 423.

Il existe un remède simple à ce type de problème : entrer toujours le nombre total de caractères en prévoyant de commencer par un ou plusieurs zéros, si nécessaire. Dans notre exemple, il aurait suffit d'entrer 0423 et 1285 pour que la situation soit nettement plus satisfaisante. Ce shéma doit vous convaincre :

0	4	2	3

1	2	8	5

4. POUR TRIER SUR UN CHAMP DE TYPE NUMERIQUE

De même que nous avons trié sur un champ de type caractère, il est possible de trier sur un champ de type numérique. Ici, les choses sont

beaucoup plus claires puisqu'alors les nombres, considérés comme tels, sont classés suivant un ordre parfaitement naturel.

A titre d'exemple, voyez comme nous avons créé un nouveau fichier PSTOCK correspondant au fichier STOCK trié sur le prix unitaire :

```
. use stock
. sort on prix to pstock
TRI TERMINE
. use pstock
. list
00001  Four à raclette 6 P  A 427    133.72     15
00002  Cafetière 12 T       A 432    137.47     32
00003  Grille pain          A 521    158.75     23
00004  Friteuse 1 kg        B 433    349.25     21
00005  Four à micro-ondes   E 248   1875.25      7
00006  Table de cuisson     E 647   2450.00      6
```

écran 9.5 : tri du fichier stock sur le prix unitaire

Remarques

1. *Tri par ordre décroissant :* dans tous nos exemples, les informations étaient triées en quelque sorte par ordre *croissant* (aussi bien les caractères pour lesquels le code ASCII allait croissant que les nombres pour lesquels la valeur allait croissant). Il s'agissait là de ''l'option par défaut'' de Dbase. Il est possible d'imposer l'ordre *décroissant* en ajoutant à la commande de tri le mot **DESCENDING.** Par exemple, pour obtenir un fichier STOCK1 trié suivant l'ordre décroissant des prix unitaires, il suffirait d'exécuter :

   SORT ON PRIX TO STOCK1 DESCENDING

2. Notez bien que *le tri ne peut s'effectuer que sur la valeur d'un champ et non sur une expression.* Ainsi, il n'est pas possible de trier le fichier stock suivant la valeur (PRIX * QTE) de chaque article en stock. Par contre, nous verrons qu'il sera possible de l'indexer suivant cette expression.

▷ **Entraînez-vous**

- Fabriquez le fichier TREPERT obtenu par tri de REPERT suivant la TAILLE. Vérifiez la structure et le contenu de TREPERT.
- Si vous ne l'avez pas encore fait en étudiant ce chapitre, créez le fichier NREPERT par
  - USE REPERT
  - SORT ON NOM TO NREPERT

- Voyez maintenant comment Dbase accepte *"d'écraser"* NREPERT par le résultat d'un autre tri :
    - USE REPERT
    - SORT ON PRENOM TO NREPERT
- Assurez vous que vous disposez de REPERT1 (double de REPERT) et voyez comme il est "facile", par maladresse, de détruire un fichier en exécutant par exemple :
    - USE STOCK
    - SORT ON PRIX TO REPERT

  Redonnez à REPERT un état "convenable" en y recopiant REPERT1
- Voyez par contre, comment Dbase refuse de "trier un fichier sur lui-même" en essayant :
    - USE REPERT
    - SORT ON NOM TO REPERT
- Voyez ce qui se passe si vous nommez un champ inexistant, par exemple :
    - USE REPERT
    - SORT ON PRIX TO PREPERT
- Ajoutez (par APPEND) dans REPERT quelques enregistrements dont le nom comporte des "caractères nationaux" (à, é, è, ù, ç...). Effectuez un tri de ce fichier sur le nom et voyez le résultat obtenu.
- Recopiez REPERT1 dans REPERT avant de passer au chapitre suivant.

X L'indexation : une (autre) façon de mettre de l'ordre et de gagner du temps

Nous avons vu que le tri fabriquait un nouveau fichier ordonné suivant les valeurs d'un certain champ. L'indexation, quant-à-elle, va se révéler être une technique « plus fine ». Elle consiste simplement à se donner le moyen de *parcourir* le fichier suivant un ordre prédéfini. Ce moyen n'est autre que la création de ce qu'on appelle une table d'index, ou plus simplement *index*. Nous verrons, qu'en outre, cet index permettra à Dbase d'accélérer la recherche d'enregistrements.

1. NOTION D'INDEX EN DBASE

Considérons notre fichier REPERT (non trié) tel que nous l'avons sauvegardé dans REPERT1 (voir sa liste en page 66). Supposez que nous disposions d'une « *table* » des noms existants, rangés par ordre alphabétique, avec, en regard, le numéro d'enregistrement correspondant :

Nom	Numéro d'enregistrement
Grillon	6
Loiseau	3
Meunier	4
Mitenne	2
Taillefert	5
Thomas	1

Dans ces conditions, vous voyez qu'il est possible de parcourir le fichier suivant l'ordre alphabétique des noms. Il suffit d'utiliser pour celà la seconde colonne de la table. Elle montre que le premier enregistrement à considérer est celui de numéro 6, le suivant porte le numéro 3 et ainsi de suite.

D'autre part, pour retrouver l'enregistrement correspondant à un nom donné, il n'est plus nécessaire d'examiner directement le fichier. Il suffit simplement de localiser le nom recherché dans la colonne de gauche et d'en déduire le numéro de l'enregistrement correspondant. Or, cette opération de localisation dans la table peut être beaucoup plus rapide que l'examen du fichier pour, au moins, deux raisons :

— cette table, plus petite que le fichier, pourra tenir intégralement en mémoire (le fichier, quant à lui, n'est amené en mémoire que par « morceaux », du moins dès qu'il atteint une certaine taille). Or, l'accès à la mémoire est beaucoup plus rapide que l'accès à la disquette !

— cette table est ordonnée. La recherche d'un nom donné peut tenir compte de cet ordre pour réduire le nombre de noms à comparer avec le nom recherché. En particulier, il est possible d'employer des techniques du type « dichotomie » ; elles ressemblent quelque peu à ce que vous faites lorsque vous cherchez un mot dans le dictionnaire (en général, vous ne lisez pas tous les mots du dictionnaire ; vous faites une approche progressive...).

Or, Dbase est effectivement capable :

1) de fabriquer un tel index : ce sera le rôle de la commande :

INDEX ON... TO... (*)

2) de l'utiliser :

— pour parcourir le fichier suivant le nouvel ordre ainsi établi (par exemple dans les commandes LIST, DISPLAY, SKIP).

(*) En anglais : indexer sur... dans...

— pour retrouver rapidement un nom donné ; ce sera le rôle de la commande :

FIND (*)

2. COMMENT CRÉER UN INDEX : INDEX ON... TO...

Considérons notre fichier REPERT tel qu'il apparaît en page... (si ce n'est pas le cas du votre, reconstituez-le à partir de REPERT1).

```
. use repert
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
. index on nom to nrepert
00006 ENREGISTREMENTS INDEXES
. list
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
```

écran 10.1 : création d'un index sur le nom pour le fichier REPERT

La commande :

INDEX ON NOM TO NREPERT

signifie : fabriquer un index à partir du champ NOM et le ranger dans un fichier nommé NREPERT. C'est ce fichier qui contiendra la table dont nous avons parlé. Notez que ce fichier sera automatiquement rangé sur la disquette(**). Il vous sera ainsi possible de l'utiliser à diverses reprises sans qu'il ne soit nécessaire de le fabriquer à nouveau. C'est ce que nous allons voir maintenant.

(*) En anglais : trouver.

(**) Notez qu'en toute rigueur, la recopie complète de l'index peut n'être effective que lorsque vous quittez Dbase ou lorsque vous changez de fichier courant (par USE). Autrement dit, là encore, il vaut mieux éviter les coupures intempestives, ou penser à refaire l'index (nous verrons comment plus loin).

3. POUR UTILISER UN INDEX EXISTANT

Nous avons fabriqué, à partir de REPERT, un fichier index nommé NREPERT. Ce n'est pas pour autant que ce dernier est systématiquement employé à chaque utilisation de REPERT. Il est nécessaire de prévenir Dbase d'utiliser cet index ; il existe pour cela 2 façons :

3.1. Nommer l'index dans la commande : USE... INDEX...

Voyez cet exemple :

```
. use repert
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
. use repert index nrepert
. list
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
```

écran 10.2 : pour utiliser un index : USE... INDEX...

Il nous montre donc deux façons de travailler avec REPERT : sans index (les enregistrements apparaissent alors dans l'ordre naturel) ; avec l'index NREPERT.

3.2. Mentionner l'index dans une commande : SET INDEX TO...

Voyez cet exemple :

```
. use repert
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
. set index to nrepert
. list
00006  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
00003  Loiseau     Albert    39400 MORREZ   84 61 28 42  1.71
00004  Meunier     Albert    75008 PARIS    49 33 27 65  1.83
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.58
00005  Taillefert  Claude    45510 SULLY    77 89 63 10  1.78
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
```

écran 10.3 : pour mentionner un index « après coup » : SET INDEX TO (*)

▷ **Entraînez-vous**

- Créez un index nommé PSTOCK sur le nom de produit du fichier STOCK.
- Quittez Dbase.
- Utilisez STOCK avec l'index PSTOCK. Listez.
- Utilisez STOCK sans index. Listez.
- Créez un index nommé TREPERT sur la taille du fichier REPERT (vous disposez donc maintenant de *deux* index pour REPERT : un sur le nom, un sur la taille).
- Voyez comme vous pouvez indifféremment utiliser *l'un ou l'autre* de ces deux fichiers index. En particulier, essayez :

  SET INDEX TO NREPERT

  SET INDEX TO TREPERT

4. LE POINTEUR SE DÉPLACE SUIVANT L'INDEX

Nous avons déjà vu que lorsque nous utilisons un index, la liste s'effectue suivant l'ordre imposé par cet index. De la même manière, la commande SKIP fait avancer de « *une position dans l'index* ».

(*) Nous verrons ultérieurement que cette commande permet également de changer d'index.

Par contre, la commande :

GOTO 1

signifie toujours : placer le pointeur sur l'enregistrement de numéro 1. *Celui-ci ne correspond généralement pas à la première position de l'index.*

Il existe une autre commande permettant de placer le pointeur sur l'enregistrement correspondant à la première position de l'index :

GOTO TOP

De la même façon :

GOTO BOTTOM

place le pointeur sur la dernière position de l'index.

Voyez ces quelques manipulations :

```
. use repert index nrepert
. list
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
. goto 2
. display
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
. skip
*** ENREGISTREMENT : 00005
. display
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
. goto top
. display
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. skip
*** ENREGISTREMENT : 00003
. display
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
. goto bottom
. display
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
```

écran 10.4 : le pointeur se déplace suivant l'index

5. POUR RETROUVER UN ENREGISTREMENT DE CLÉ DONNÉE : FIND

Comme nous vous l'avons laissé entrevoir dans l'introduction, l'usage d'un index va permettre à Dbase de retrouver rapidement un enregistrement dont on lui fournit la clé.

5.1. La commande FIND

Voyez cet exemple d'emploi de la commande :

FIND

```
. use repert index nrepert
. find Mitenne
. display
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
```

écran 10.5 : pour retrouver un enregistrement de clé donnée

Voyez comme cette commande se contente de placer le pointeur sur le premier enregistrement contenant la clé cherchée. Elle ne l'affiche pas, d'où la nécessité d'utiliser ensuite DISPLAY. D'autre part, si plusieurs enregistrements possèdent la même clé, il faudra exécuter une ou plusieurs commandes SKIP pour y accéder. En voici un exemple :

```
. use repert
. index on ville to vrepert
00006 ENREGISTREMENTS INDEXES
. list
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
. find PARIS
. display
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
. skip
*** ENREGISTREMENT : 00006
. display
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. skip
*** ENREGISTREMENT : 00005
. display
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
```

écran 10.6 : quand plusieurs enregistrements ont la même clé

Remarques :

Dans ce deuxième exemple, nous avons créé pour REPERT un autre index sur le nom de ville nommé VREPERT. Voyez également comme la commande INDEX qui crée cet index établit en même temps le « lien » avec lui, de sorte qu'il n'est pas utile, à ce niveau, d'exécuter une commande telle que :

USE REPERT INDEX VREPERT

5.2. Quand la clé recherchée n'existe pas dans l'index

Dbase le signale simplement comme dans cet exemple :

```
. use repert index nrepert
. find Jules
*** ENREGISTREMENT NON TROUVE ***
. find THOMAS
*** ENREGISTREMENT NON TROUVE ***
```

écran 10.7 : quand la clé n'existe pas

5.3. Quand on abrège la clé

FIND recherche dans l'index associé au fichier une clé *égale* à la valeur que vous lui avez fournie. Or, comme nous l'avons déjà dit, en Dbase les comparaisons d'égalité peuvent être partielles ou exactes suivant la valeur de l'indicateur nommé EXACT (*).

```
. use repert index nrepert
. set exact off
. find Mitenne
. display
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
. find Mite
. display
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
. set exact on
. find Meunier
. display
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
. find Meu
*** ENREGISTREMENT NON TROUVE ***
```

écran 10.8 : clé exacte ou partielle

(*) Par défaut, au lancement de Dbase, celui-ci est désactivé (OFF).

5.4. Attention à l'écriture de la clé

Nous pouvons indifféremment taper les commandes Dbase en majuscules ou en minuscules (ou même en « panachant » majuscules et minuscules). Il en allait de même pour les noms de fichiers ou de champs.

Par contre, il n'en va plus de même en ce qui concerne la clé que l'on fait rechercher par FIND. Dans ce cas, en effet, il est nécessaire de respecter exactement l'écriture de l'information telle qu'elle figure dans le fichier.

Pour les champs de type numérique toutefois, une tolérance a lieu concernant d'éventuels zéros de fin situés après le point décimal. Ainsi Dbase reconnaît 1.70 égal à 1.7 (même lorsque l'indicateur EXACT est actif).

Voyez ces quelques exemples :

```
. use repert index nrepert
. find TAILLEFERT
*** ENREGISTREMENT NON TROUVE ***
.
. index on taille to trepert
00006 ENREGISTREMENTS INDEXES
. find 1.5
*** ENREGISTREMENT NON TROUVE ***
. find 1.580
. display
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
```

écran 10.9 : bien écrire ce que l'on cherche par FIND

Remarque :

Il est possible, dans la commande FIND, de placer entre guillements la clé recherchée. Par exemple, ces deux commandes sont équivalentes :

FIND Taillefert

FIND ''Taillefert''

6. FIND ou LOCATE ?

Lorsque, par exemple, vous utilisez REPERT avec l'index NREPERT, il peut vous arriver d'avoir besoin de retrouver une personne habitant une ville donnée. Plusieurs possibilités s'offrent alors à vous.

S'il existe un index sur la ville, vous pouvez le faire utiliser par Dbase par une commande SET INDEX, puis rechercher votre enregistrement par FIND (et éventuellement SKIP).

S'il n'existe pas d'index sur la ville (et que vous ne souhaitez pas le créer), il vous est toujours possible d'effectuer une recherche par LOCATE. Vous procédez pour celà de la même manière que si aucun index n'avait été associé au fichier.

```
. use repert index nrepert
. find Grillon
. display
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. locate for ville = "PARIS"
*** ENREGISTREMENT : 00006
. display
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
. continue
*** ENREGISTREMENT : 00004
. display
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
```

écran 10.10 : FIND ou LOCATE.

Remarque :

Il est toujours possible d'employer LOCATE, au lieu de FIND pour retrouver un enregistrement de clé donnée. Cependant, dans ce cas, la recherche est moins rapide, de sorte que cette méthode ne présente aucun intérêt.

▷ **Entraînez-vous**

Assurez-vous que vous disposez bien du fichier REPERT tel qu'il apparaît en page... (ou dans REPERT1) et des deux fichiers index correspondants : NREPERT et TREPERT.

- Voyez ce qui se produit lorsque vous cherchez à exécuter une commande FIND sur un fichier auquel aucun index n'a été associé, comme dans cet exemple :
  - USE REPERT
  - FIND Mitenne

- Créez un index sur le nom de ville par :
  - USE REPERT
  - INDEX ON VILLE TO VREPERT
- En utilisant, lorsque c'est possible, l'index approprié, recherchez successivement les enregistrements correspondants :
  - à la ville : SULLY
  - au prénom : Laurence
  - à la taille : 1,78
  - au nom : Grillon
- Effectuez les mêmes recherches sur des valeurs « abrégées » en activant ou en désactivant l'indicateur EXACT. Voyez comme son état est sans effet sur les recherches portant sur des valeurs numériques.
- En associant à REPERT l'index VREPERT (par exemple par USE REPERT INDEX VREPERT) et *sans changer d'index*, effectuez les mêmes recherches que précédemment (SULLY, Laurence, 1.78, Grillon).
- En utilisant d'abord l'index TREPERT puis VREPERT, trouvez toutes les personnes dont la taille est comprise entre 1,65 m et 1,75 m.

7. QUAND L'INDEX EST AUTOMATIQUEMENT MIS À JOUR

Jusqu'ici nous nous sommes contentés de consulter un fichier auquel était associé un index. Mais nous pouvons être amenés à effectuer des mises à jour d'un tel fichier (modifications, ajout, effacement). Dans ce cas, Dbase **« actualise » automatiquement l'index** qui est utilisé avec le fichier (c'est-à-dire l'index qui a été nommé dans une commande USE ou SET INDEX).

Prenons un exemple pour illustrer ce que représente une actualisation d'un index. Supposons que nous utilisons REPERT associé à l'index NREPERT et que ce dernier corresponde à cette table :

Grillon	6
Loiseau	3
Meunier	4
Mitenne	2
Taillefert	5
Thomas	1

Supposons que nous remplacions (par EDIT), dans l'enregistrement numéro 4, le nom Taillefert par Jolibois. Dbase va alors automatiquement reporter cette modification dans l'index NREPERT. En outre, il va le réorganiser pour qu'il soit convenablement ordonné sur la clé. NREPERT se présentera alors ainsi :

Grillon	6
Jolibois	5
Loiseau	3
Meunier	4
Mitenne	2
Thomas	1

Remarque : sur des fichiers de quelque importance, la réorganisation systématique d'index peut se révéler un peu longue. Aussi, lorsque l'on doit effectuer beaucoup de « mises à jour » sur un gros fichier, il peut être préférable de le faire sur un fichier non indexé. On reconstitue l'index ultérieurement soit par la commande INDEX ON..., soit par la commande REINDEX que nous étudierons dans le prochain chapitre.

▷ **Entraînez-vous**

- Ouvrez votre fichier REPERT en lui associant l'index NREPERT par :
  USE REPERT INDEX NREPERT
- Remplacez, par EDIT, le nom Taillefert par Jolibois. Listez votre fichier et voyez comme l'index a été convenablement actualisé.
- Redonnez à REPERT et NREPERT leur « état initial » par :
  - USE REPERT1
  - COPY TO REPERT
  - USE REPERT
  - INDEX ON NOM TO NREPERT

Si vous n'êtes pas certain que votre index PSTOCK correspond bien à votre fichier STOCK, recréez le ainsi :

- USE STOCK
- INDEX ON PRODUIT TO PSTOCK

- Ouvrez votre fichier STOCK en lui associant l'index PSTOCK par :
  USE STOCK INDEX PSTOCK.
- Ajoutez plusieurs enregistrements de votre choix par APPEND.
- Vérifiez à l'aide d'une liste que l'index a été considérablement réorganisé.

- Procédez à des modifications par EDIT et vérifiez de même.
- Effacez tout ou partie des enregistrements ainsi ajoutés. Vérifiez la réorganisation de l'index.
- Voyez ce qui se passe si vous cherchez à utiliser un fichier avec un index « mal assorti », par exemple comme ceci :

  USE STOCK INDEX NREPERT.
- Avant d'aborder la suite, redonnez leur « état initial » à REPERT et STOCK en y recopiant REPERT1 et STOCK1.

XI Pour en savoir plus sur l'indexation

Le chapitre précédent vous a présenté les bases de cette importante technique qu'est l'indexation. Ici, nous vous donnons des compléments d'information qui se révèleront très utiles lorsque vous aborderez vos propres réalisations.

Il est tout à fait possible, dans une première étude, de « sauter » ou de « survoler » ce chapitre.

1. POUR QUE DBASE RÉORGANISE AUTOMATIQUEMENT PLUSIEURS INDEX

Nous avons été amenés à *créer* plusieurs index pour le même fichier REPERT, à savoir : NREPERT, VREPERT et TREPERT. Par contre, nous *n'utilisions qu'un seul index à la fois.* Dans ces conditions, vous voyez qu'un problème de réorganisation des index apparaît en cas de mise à jour du fichier REPERT.

En effet, supposons, par exemple, qu'à un instant donné, nous ayons selectionné l'index NREPERT par :

- USE REPERT INDEX NREPERT

Si nous modifions le *nom* et la *ville* d'un enregistrement, il y aura automatiquement réorganisation de l'index ''courant'', à savoir NREPERT. Par contre, l'index VREPERT ne sera pas modifié. Il ne correspondra donc plus au fichier REPERT puisqu'il comportera un nom de ville différent (et qu'en plus, cet élément ne sera plus à sa place).

Bien entendu, dans de telles circonstances, il est toujours possible de fabriquer à nouveau notre fichier VREPERT, soit par INDEX, soit par REINDEX. Mais, il est également possible de demander à Dbase **d'actualiser simultanément plusieurs index.** Il suffit pour celà de les mentionner dans la commande USE (ou SET INDEX) à la suite de **l'index courant** qui **continuera à être le seul utilisé pour la recherche.**

Dans notre exemple, il nous suffira d'établir ainsi le lien avec REPERT :

- USE REPERT INDEX NREPERT, VREPERT, TREPERT

pour que toute mise à jour du fichier REPERT soit convenablement répercutée (s'il y a lieu) dans nos trois index.

Remarques :

1) Bien que l'on nomme plusieurs index, ce n'est que le premier qui sert à la recherche. Ainsi, dans notre exemple, une commande telle que FIND PARIS n'aboutira pas (à moins que vous n'ayez dans votre fichier un individu nommé PARIS !). Notez, au passage, que l'ordre d'énumération des index, à partir du deuxième, est sans importance.

2) Si vous modifiez l'index courant par SET INDEX, il faudra, comme dans USE, nommer à la suite les index que vous souhaitez conserver en actualisation automatique. Par exemple, pour que l'index courant devienne TREPERT (au lieu de NREPERT), vous utiliserez :

   - SET INDEX TO TREPERT, NREPERT, VREPERT.

▷ **Entraînez-vous**

Assurez-vous que vous disposez de REPERT tel qu'il figure dans REPERT1 et des trois index correspondants : NREPERT, VREPERT et TREPERT.

- Ouvrez votre fichier REPERT en lui associant ces trois index par :
  - USE REPERT INDEX NREPERT, VREPERT, TREPERT.
- Ajoutez plusieurs enregistrements de votre choix par APPEND.
- Vérifiez, par des listes que les trois index ont été convenablement réorganisés.
- Procédez de même avec des modifications et des effacements.

2. POUR METTRE À JOUR DES INDEX

Il peut vous arriver de ne pas faire effectuer par Dbase la mise à jour automatique de vos index :

- soit par oubli
- soit pour gagner du temps (les réorganisations d'index pouvant devenir franchement intolérables sur des fichiers assez importants).

Bien entendu, il est toujours possible de recréer *totalement* ces index à l'aide de la commande INDEX ON... TO... Mais vous pouvez également obtenir le même résultat et plus rapidement avec la commande :

REINDEX (*)

Celle-ci *actualise tous les index associés au fichier courant.* En voici un exemple :

```
. use repert index nrepert, vrepert, trepert
. reindex

MISE A JOUR DE L'INDEX: A:NREPERT .NDX
00006 ENREGISTREMENTS INDEXES

MISE A JOUR DE L'INDEX: A:VREPERT .NDX
00006 ENREGISTREMENTS INDEXES

MISE A JOUR DE L'INDEX: A:TREPERT .NDX
00006 ENREGISTREMENTS INDEXES
```

écran 11.1 : pour réorganiser ses index.

Voyez comme Dbase vous informe à la fois des fichiers INDEX qu'il a cherché à réorganiser (même s'il n'y a opéré aucune modification) et du nombre total d'éléments y figurant.

3. QUAND LA CLÉ EST UNE EXPRESSION

Jusqu'ici, nous avons créé nos index en utilisant comme clé un nom de champ. C'est là l'utilisation la plus courante. Néanmoins, Dbase

(*) En anglais : réindexer.

vous autorise à créer un index à partir d'une *expression quelconque*. Voyons en deux exemples.

3.1. Pour indexer REPERT sur la VILLE et le NOM

Nous avons déjà créé un index VREPERT sur les noms de ville. Mais, lorsque plusieurs enregistrements comportent la même ville, ils se trouvent rangés dans l'index suivant leur ordre naturel dans le fichier. Nous pourrions souhaiter qu'ils apparaissent ordonnés par exemple suivant le nom. (Attention, il ne s'agit pas de créer un index sur les noms ; le classement doit s'opérer sur la ville, le nom servant uniquement en quelque sorte à ''départager les ex aequo'').

Nous pouvons parvenir à ce résultat en choisissant comme clé d'index la chaîne obtenue en concaténant (mettant ''bout à bout'') les deux champs VILLE et NOM. ''L'expression'' correspondante n'est rien d'autre que :

VILLE + NOM

d'où la commande de création d'index :

INDEX ON VILLE + NOM TO VNREPERT

```
. use repert
. index on ville+nom to vnrepert
00006 ENREGISTREMENTS INDEXES
. list
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
. find PARIS Meunier
*** ENREGISTREMENT NON TROUVE ***
. find PARIS   Meunier
. display
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
```

écran 11.2 : pour indexer sur ville et nom.

Voyez comme lors de la recherche par FIND, il est nécessaire de prévoir les *éventuels espaces* figurant à la fin du champ NOM.

Remarque :
Ne confondez pas cette création d'un index sur la clé VILLE + NOM avec la création des deux index : NREPERT et VREPERT.

3.2. Pour indexer REPERT sur le premier caractère du NOM et les deux premiers caractères de la VILLE

L'expression clé n'est rien d'autre que :

$(NOM,1,1) + $(VILLE,1,2).

```
. use repert
. index on $(nom,1,1)+$(ville,1,2) to nvrepert
00006 ENREGISTREMENTS INDEXES
. list
00006  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
00003  Loiseau    Albert   39400 MORREZ   84 61 28 42  1.71
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00005  Taillefert Claude   45510 SULLY    77 89 63 10  1.78
. find MFR
. display
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.58
. find MPA
. display
00004  Meunier    Albert   75008 PARIS    49 33 27 65  1.83
```

écran 11.3 : pour indexer sur le début du nom et le début de la ville

A titre indicatif, l'index NVREPERT serait constitué ainsi :

Clé	Numéro d'enregistrement
GPA	6
LMO	3
MFR	2
MPA	4
TNE	1
TSU	5

Remarque : L'avantage des ''clés courtes'' apparaît surtout pour des fichiers importants, au niveau du gain de place (disquette) et du gain de temps de recherche.

4. POUR SAVOIR OÙ VOUS EN ÊTES

Lorsque vous manipulez beaucoup de fichiers avec plusieurs index, il peut vous arriver de vous sentir quelque peu perdu. Notamment, vous risquez d'être amenés à vous poser des questions telles que :

— Quels sont les fichiers index dont je dispose ?

— Quels fichiers index sont *actuellement* associés à mon fichier courant ?

— A quel fichier de données correspond un certain fichier index ?

Nous allons voir que deux commandes : LIST FILES et DISPLAY STATUS permettent de répondre à ces questions (partiellement pour la troisième).

4.1. Pour connaître tous les fichiers index

Nous avons vu que la commande LIST FILES nous fournit la liste des fichiers de ''type'' DBF et uniquement ceux-ci. Or, les fichiers index créés par Dbase sont du type **NDX**, de sorte qu'ils n'apparaissent pas avec cette simple commande. Comment les connaître ? Il faut en fait utiliser :

LIST FILES LIKE... (*)

Voyez cet exemple :

```
display files like *.ndx

NREPERT .NDX        VREPERT .NDX        TREPERT .NDX        VNREPERT.NDX
NVREPERT.NDX
```

écran 11.4 : pour lister les noms des fichiers index

La commande :

LIST FILES LIKE *.NDX

signifie : lister les noms de tous les fichiers de la forme : *.NDX. En fait * est une sorte de ''joker'' représentant n'importe quelle suite de

(*) En anglais LIKE signifie : comme, semblable à.

caractères. Cette commande nous affiche donc tous les noms des fichiers ayant un nom quelconque et un type NDX.

Remarque :

LIST FILES LIKE REPERT.* afficherait les fichiers ayant comme nom REPERT et de type quelconque.

LIST FILES LIKE *.* afficherait les noms de tous les fichiers, quel que soit leur type.

4.2. Pour connaître "l'état" de votre travail : DISPLAY STATUS.

Voici un exemple des informations fournies par la commande :

DISPLAY STATUS

```
. use repert index nrepert, nvrepert, vnrepert
. display status

BASE ACTIVE        - A:REPERT  .DBF
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION

 INDEX:             EXPRESSION CLE:
 A:NREPERT .NDX     nom
 A:NVREPERT.NDX     $(nom,1,1)+$(ville,1,2)
 A:VNREPERT.NDX     ville+nom

 ->

 DATE DU JOUR        - 12/06/86
 UNITE DE DEFAUT     - A:
 ALTERNATE - OFF     BELL       - ON
 CARRY     - OFF     COLON      - ON
 CONFIRM   - OFF     CONSOLE    - ON
 DEBUG     - OFF     DELETE     - OFF
 ECHO      - OFF     EJECT      - ON
 ESCAPE    - ON      EXACT      - OFF
 INTENSITY - ON      LINKAGE    - OFF
 PRINT     - ON      RAW        - OFF
 STEP      - OFF     TALK       - ON
```

écran 11.5 : la commande DISPLAY STATUS

Vous voyez que cette commande vous rappelle le nom du fichier courant (nommé ici « base active »). Elle vous fournit le nom des fichiers index utilisés, ainsi que l'expression clé qui a servi à les constituer.

Notez bien que cette commande ne fournit de l'information que pour

les fichiers index effectivement utilisés. En aucun cas, elle ne permet de retrouver *tous* les fichiers index constitués à partir d'un fichier donné.

D'autre part, après vous avoir fourni des informations concernant la zone de travail, cette commande peut également vous fournir l'état de tous les ''indicateurs''. Il vous suffit pour cela de taper ''RETURN'' (ce qui vous est rappelé par le symbole – >).

4.3. Comment éviter d'employer un ''mauvais'' index

Tout d'abord, si vous cherchez à utiliser un fichier index qui n'a pas été constitué à partir du fichier courant, il existe deux comportements possibles de Dbase :

a) le fichier index en question possède une expression clé *incorrecte* par rapport au fichier courant. C'est ce qui se produirait par exemple avec :

USE REPERT INDEX PSTOCK

Dans ce cas, Dbase le signale clairement.

b) le fichier index en question possède une expression clé qui (par malchance) est *correcte* par rapport au fichier courant. C'est là un phénomène qui ne peut se produire que lorsque deux fichiers différents comportent des champs de même nom. Dans ce cas, Dbase **accepte** la commande.

Comme par ailleurs, vous ne disposez d'aucun moyen direct de connaître les fichiers index correspondants à un fichier donné, vous voyez l'intérêt qu'il y a :

— de choisir judicieusement vos noms de fichiers, y compris les index. Evitez pour ces derniers des noms tels que INDEX1, INDEX, INDEXA, X,... (de toutes façons, il est inutile de dire dans le nom d'un fichier index qu'il s'agit d'un index puisqu'il sera nécessairement identifiable par son type : NDX)

— d'éviter, dans la mesure du possible, d'avoir des champs de même nom dans des fichiers différents. (Ce faisant, vous ne risquerez plus d'associer par erreur à un fichier un index destiné à un autre).

Par contre, il existe une situation contre laquelle il n'y a pas d'autre façon de se protéger que la prudence. Il s'agit du cas où vous cherchez à utiliser un index qui correspond bien au fichier mais qui n'a pas été actualisé (suite à des modifications apportées sans l'usage de l'index). Dans ce cas, les conséquences sont multiples : liste mal orga-

nisée ; recherche infructueuse car FIND continue à examiner les valeurs figurant dans l'index et non celles du fichier ; enregistrement que l'on ne peut pas situer car l'index ''pointe'' en dehors du fichier actuel, etc... Quoiqu'il en soit, un seul remède efficace s'offre à vous : reconstituer l'index.

▷ Entraînez-vous

Assurez vous que vous disposez de REPERT tel qu'il figure dans REPERT1 et de l'index correspondant NREPERT.

- Ouvrez REPERT *sans index* par :

  USE REPERT

- Remplacez, par EDIT, le nom Taillefert par Baillefert. Vérifiez par LIST.

- Utilisez l'index NREPERT, par exemple par :

  USE REPERT INDEX NREPERT

puis listez votre fichier et voyez où se situe l'enregistrement modifié.

- Essayez :

  FIND Taillefert
  FIND Baillefert

Expliquez le phénomène.

- Reconstituez REPERT dans son état initial (par exemple en y recopiant REPERT1). Ouvrez-le à nouveau sans index et ajoutez trois enregistrements par APPEND. Utilisez l'index NREPERT et cherchez à lister votre fichier.

XII Édition de rapports

Vous savez déjà comment extraire de l'information de vos fichiers, grâce notamment à des commandes telles que LIST ou DISPLAY. Mais Dbase vous permet de mettre vos informations un peu plus en forme en réalisant ce que l'on nomme des rapports. La commande REPORT va vous permettre de sélectionner les informations qui vous intéressent, d'en définir la présentation et même de réaliser certains calculs.

1. CRÉATION D'UN RAPPORT SIMPLIFIÉ : LA COMMANDE REPORT

La création d'un rapport est facile à mettre en œuvre mais elle fait intervenir quelques notions nouvelles (comme le *fichier format,* les *totaux* et *sous-totaux*). Pour bien vous familiariser avec son usage, nous allons commencer par créer un "rapport" assez simpliste.

Nous supposons que notre fichier STOCK se présente ainsi :

```
. list
00001  Cafetière 12 T        A 432    137.47     32
00002  Four à micro-ondes    E 248   1875.25      7
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
00005  Friteuse 1 kg         B 433    349.25     21
00006  Table de cuisson      E 647   2450.00      6
```

Nous allons réaliser un rapport comportant pour chaque produit le *libellé* et la *quantité.* Nous exécutons pour cela la commande :

REPORT (*)

et nous constatons que Dbase nous pose beaucoup de questions (voir écran 12.1)

Tout d'abord il nous demande le nom de ce qu'il appelle un ''fichier de génération d'état''. Il ne s'agit pas d'un fichier destiné à recevoir le rapport souhaité mais simplement les ''directives'' permettant de générer ce rapport, directives que nous allons exprimer par des réponses aux questions que va nous poser Dbase.

La question relative aux ''options'' concerne la ''pagination'' sur imprimante. Nous conservons les valeurs par défaut procurées par Dbase en frappant directement ''RETURN''.

Nous choisissons ensuite de ne pas faire précéder notre rapport d'une en-tête et d'utiliser un simple interligne. Nous avons également choisi de ne pas afficher de totaux (nous verrons ce que sont ces totaux dans l'exemple suivant).

Enfin vient la partie la plus importante avec la question : COLONNE LONGUEUR, CONTENU. En effet, votre rapport, au même titre qu'une liste fournie par LIST, va se présenter comme un *assemblage de colonnes.* Pour chaque colonne qu'il repère par un numéro (qui ne figurera pas sur le rapport), Dbase vous demande de fournir la *''longueur''* (sa taille) et le *''contenu''*, c'est-à-dire l'expression que vous voulez y voir affichée. Il vous demande également le *titre* qui apparaîtra en haut de cette colonne.

(*) En anglais : établir un état, un rapport.

```
. report
DONNEZ LE NOM DU FICHIER DE GENERATION D'ETAT : etatstoc
OPTIONS  M=MARGE GAUCHE, L=LIGNES/PAGE, W=LARGEUR DE PAGE
DESIREZ-VOUS UN ENTETE (Y/N) ? n
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? n
DESIREZ-VOUS DES TOTAUX (Y/N) ? n
COLONNE LONGUEUR,CONTENU
001      25,produit
DONNEZ LE TITRE : Désignation Produit
002      20,qte
DONNEZ LE TITRE : Quantité en stock
003
```

```
 PAGE N. 00001
17/06/86


   Désignation Produit        Quantité en stock

Cafetière 12 T                              32
Four à micro-ondes                           7
Grille pain                                 23
Four à raclette 6 P                         15
Friteuse 1 kg                               21
Table de cuisson                             6
```

écran 12.1 : définition et affichage d'un rapport

Ici, nous décrivons nos deux colonnes et pour signaler que nous n'en souhaitons pas d'autres, nous frappons directement ''return'' lorsque Dbase nous demande de décrire la troisième (en affichant 003).

La phase de définition du ''fichier de génération d'état'' s'achève là et Dbase va maintenant réaliser le rapport (ou état) proprement dit. Il apparaît comme indiqué dans la seconde partie de l'écran 12.1. Notez la présence d'un numéro de page et de la date courante (celle fournie lors du lancement de Dbase)

2. LE FICHIER ''FORMAT''

Lors de l'exécution de la commande REPORT, la première question posée par Dbase concernait le nom du ''fichier de génération d'état'' (qu'on apelle aussi ''fichier format''). Comme nous vous l'avons annoncé, ce fichier est destiné à contenir les directives nécessaires à la réalisation du raport (ou ''état'').

Ce fichier a été *créé* par la commande REPORT avec l'extension FRM. Vérifions le à l'aide de la commande :

LIST FILES LIKE

```
. list files like *.frm

ETATSTOC.FRM
```

écran 12.2 : le fichier ''format'' créé par REPORT

Si maintenant nous exécutons à nouveau la commande REPORT avec le même fichier format, Dbase va réaliser directement notre rapport sans nous poser à nouveau les questions. Vérifions le :

```
. report
DONNEZ LE NOM DU FICHIER DE GENERATION D'ETAT : etatstoc
```

```
 PAGE N. 00001
17/06/86



   Désignation Produit        Quantité en stock

Cafetière 12 T                              32
Four à micro-ondes                           7
Grille pain                                 23
Four à raclette 6 P                         15
Friteuse 1 kg                               21
Table de cuisson                             6
```

écran 12.3 : édition d'un rapport à partir d'un fichier format existant

Remarques :

1) il est possible de mentionner directement le fichier ''format'' dans la commande REPORT au lieu de laisser Dbase nous demander son nom. Il suffit d'écrire, par exemple :

REPORT FORM ETATSTOC

2) *Ici,* la commande :

LIST PRODUIT, QTE

nous aurait fourni un résultat assez proche de notre rapport. Néanmoins, nous n'aurions pas pu choisir la largeur des colonnes, ni leur donner un titre. En outre, nous n'aurions pas disposé de l'option TO PRINT (que nous vous exposerons dans le prochain ''Entraînez-vous'').

3. RAPPORT AVEC SÉLECTION

Dans la commande REPORT, il est possible de demander à Dbase de ne prendre en compte que certains enregistrements, à l'aide d'une condition introduite (classiquement) par FOR. Celà s'applique aussi bien lorsque l'on emploie un fichier format existant que lorsqu'il faut le créer. En voici un exemple, dans lequel nous utilisons le fichier format créé précédemment pour établir un rapport ne concernant que les articles de catégorie A.

```
. report form etatstoc for cat="A"
```

```
 PAGE N. 00001
17/06/86


   Désignation Produit        Quantité en stock

Cafetière 12 T                                32
Grille pain                                   23
Four à raclette 6 P                           15
```

écran 12.4 : rapport avec sélection

▷ **Entraînez-vous**

- Voyez comme il est facile d'éditer un rapport sur imprimante en ajoutant l'option TO PRINT dans la commande REPORT. Essayez, par exemple :
  - USE STOCK
  - REPORT FORM ETATSTOC TO PRINT
- Voyez ce qui se passe si vous essayez d'éditer un rapport à partir d'un fichier format non adapté au fichier courant, comme dans cet exemple :
  - USE REPERT
  - REPORT FORM ETATSTOC

- Créez un fichier format, nommé TAILLES permettant d'établir, à partir d'un fichier tel que REPERT, un rapport se présentant ainsi :

```
 PAGE N.  00001
17/06/86


    Je m'appelle          Je mesure

Thomas                          1.75
Mitenne                         1.58
Loiseau                         1.71
Meunier                         1.83
Taillefert                      1.78
Grillon                         1.68
```

- Utilisez le même fichier TAILLES pour réaliser un rapport semblable pour les seuls habitants de PARIS.
- Effectuez un tri sur la quantité en stock du fichier STOCK dans STOCKT. Réalisez un rapport à partir de STOCKT en employant le format fichier ETATSTOC.

  Comparez le résultat ainsi obtenu avec notre rapport de la page...
- Voyez comme vous obtiendriez le même résultat en utilisant un index :
  - USE STOCK
  - INDEX ON QTE TO QSTOCK
- Recopiez le fichier STOCK dans STOCKBIS ; ajoutez quelques enregistrements de votre choix à STOCKBIS. Réalisez un rapport à partir de STOCKBIS en employant toujours le même fichier format.

4. POUR FAIRE DES CALCULS DANS LES RAPPORTS : EXPRESSIONS ET TOTAUX

Dans les exemples que nous avons recontrés, le contenu des colonnes correspondait simplement à la valeur d'un champ (PRODUIT, QTE,...). En fait, Dbase vous permet d'y placer n'importe quelle **expression**.

D'autre part, il est possible de demander d'effectuer des "**totaux**" pour tout ou partie des colonnes de type numérique.

Voici un exemple, correspondant au fichier STOCK, dans lequel nous établissons un rapport comportant 4 colonnes, dans lesquelles nous plaçons :

— les 8 premiers caractères du libellé, avec l'expression :

$(PRODUIT, 1,8)

— La catégorie et la référence, mises bout-à-bout, avec l'expression :

CAT + REF

— Le nombre d'articles correspondant : QTE

— La valeur correspondante, avec l'expression :

PRIX * QTE

```
. use stock
. report form rapstock
OPTIONS  M=MARGE GAUCHE, L=LIGNES/PAGE, W=LARGEUR DE PAGE
DESIREZ-VOUS UN ENTETE (Y/N) ? n
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? n
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
DESIREZ-VOUS DES SOUS-TOTAUX DANS VOTRE RAPPORT (Y/N) ? n
COLONNE LONGUEUR,CONTENU
001      8,$(produit,1,8)
DONNEZ LE TITRE : PRODUIT
002      5,cat+ref
DONNEZ LE TITRE : CODE
003      9,qte
DONNEZ LE TITRE : EN STOCK
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
004      10,qte*prix
DONNEZ LE TITRE : VALEUR
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
005
```

```
          PAGE N. 00001
         18/06/86


          PRODUIT   CODE   EN STOCK    VALEUR

         Cafetièr A432          32     4399.04
         Four à m E248           7    13126.75
         Grille p A521          23     3651.25
         Four à r A427          15     2005.80
         Friteuse B433          21     7334.25
         Table de E647           6    14700.00

** TOTAL **
                               104    45217.09
```

écran 12.5 : rapport avec expressions et totaux

Notez qu'à partir du moment où nous demandons des totaux, Dbase nous demande si nous souhaitons également des ''sous-totaux''. Ici, nous avons répondu **N.**

Voyez également comme, ayant dit que nous désirions des totaux, Dbase nous demande de le ''confirmer'' individuellement pour chaque colonne renfermant des informations de type numérique.

▷ **Entraînez-vous**

- Utilisez le fichier format que nous venons de créer pour établir un rapport ne concernant que les articles de catégorie B.
- Établissez un rapport, à partir du fichier REPERT, comportant les informations suivantes :
  - Nom
  - Les deux premiers caractères du prénom
  - Le numéro de département (seul)
  - Les six derniers chiffres du numéro de téléphone.

5. POUR OBTENIR DES ''TOTAUX PARTIELS''

Dbase vous permet d'obtenir, en plus des totaux, ce qu'il appelle des ''**sous-totaux**''. Il s'agit de totaux partiels réalisés pour les colonnes faisant déjà l'objet de totaux et portant sur des enregistrements consécutifs ayant une valeur commune : le nom, la ville, la catégorie, etc... Mais tout cela n'a de véritable intérêt que si le fichier est trié ou indexé suivant le champ correspondant.

A titre d'exemple, complétons notre rapport de la page 107 en prévoyant des sous-totaux par catégorie.

```
. use stock
. index on cat to cstock
00006 ENREGISTREMENTS INDEXES
. report form rapstoc1
OPTIONS  M=MARGE GAUCHE, L=LIGNES/PAGE, W=LARGEUR DE PAGE
DESIREZ-VOUS UN ENTETE (Y/N) ? y
DONNEZ L'ENTETE : ETAT DU STOCK PAR CATEGORIE
DESIREZ-VOUS UN DOUBLE INTERLIGNE (Y/N) ? n
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
DESIREZ-VOUS DES SOUS-TOTAUX DANS VOTRE RAPPORT (Y/N) ? y
DONNEZ LE CHAMP DE SOUS-TOTAL : cat
DESIREZ-VOUS UNIQUEMENT UN RESUME DU RAPPORT (Y/N) ? n
CHANGEMENT DE PAGE APRES LES SOUS-TOTAUX (Y/N) ? n
ENTREZ LE TITRE POUR LES SOUS-TOTAUX : CATEGORIE
COLONNE LONGUEUR,CONTENU
001      8,$(produit,1,8)
DONNEZ LE TITRE : PRODUIT
002      5,cat+ref
DONNEZ LE TITRE : CODE
003      9,qte
DONNEZ LE TITRE : EN STOCK
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
004      10,qte*prix
DONNEZ LE TITRE : VALEUR
DESIREZ-VOUS DES TOTAUX (Y/N) ? y
005
```

écran 12.6 : définition d'un rapport avec sous-totaux par catégorie

```
 PAGE N. 00001
18/06/86

                                ETAT DU STOCK PAR CATEGORIE


 PRODUIT   CODE   EN STOCK    VALEUR

* CATEGORIE A
Cafetièr A432          32     4399.04
Grille p A521          23     3651.25
Four à r A427          15     2005.80

** SOUS-TOTAL **
                       70    10056.09


* CATEGORIE B
Friteuse B433          21     7334.25

** SOUS-TOTAL **
                       21     7334.25


* CATEGORIE E
Four à m E248           7    13126.75
Table de E647           6    14700.00

** SOUS-TOTAL **
                       13    27826.75


** TOTAL **
                      104    45217.09
```

écran 12.7 : le rapport obtenu (correspond à l'écran 12.6)

Voyez comme le fait de demander des sous-totaux amène Dbase à vous faire préciser le champ correspondant (Ne confondez pas ce champ avec les colonnes faisant l'objet de totaux et donc de sous-totaux). Il vous demande également si vous souhaitez seulement un ''résumé'' ; vous en trouverez un exemple dans le prochain ''Entraînez-vous''. Enfin vous pouvez mentionner un titre qui apparaîtra avant la liste correspondant à chaque valeur du champ de sous-total.

Voyez comme nous avons pris soin d'indexer notre fichier STOCK suivant la catégorie.

▷ **Entraînez-vous**

- Voyez ce que produit le fichier format CSTOCK lorsque vous l'appliquez à un fichier STOCK non indexé :

  USE STOCK

  REPORT FORM CSTOCK

- Voyez ce que produit un "résumé du rapport" en créant un nouveau fichier format concernant STOCK en fournissant les mêmes réponses que précédemment, exception faite pour la question relative au résumé.

XIII Pour modifier la structure d'un fichier

Il peut vous arriver de découvrir, alors que votre fichier a été constitué, que la structure que vous lui aviez attribuée ne vous convient plus. Voici quelques situations que vous pouvez rencontrer :

— Vous n'avez pas été suffisamment généreux dans le choix de la taille d'un champ de sorte que vous êtes obligé de ''tronquer'' certaines informations. Vous aimeriez pouvoir *augmenter la taille du champ* en question.

— au contraire, vous avez prévu trop de place pour un champ et vous êtes gêné par le fait qu'alors votre fichier occupe plus de place que nécessaire sur la disquette. Vous aimeriez pouvoir *réduire la taille du champ* en question.

— il s'avère qu'un *nouveau champ* est nécessaire

— au contraire, un champ est devenu inutile et vous aimeriez pouvoir le *supprimer.*

— Vous aimeriez *modifier le nom* que vous avez attribué à un champ.

La plupart du temps, il sera nécessaire de commencer par modifier la structure de votre fichier (nous verrons qu'il y a une exception). Ce sera le rôle de la commande MODIFY STRUCTURE. Mais, comme nous allons le voir, cette commande détruit toutes les données du

fichier. Il sera donc nécessaire, au préalable, d'effectuer une copie à l'aide de la commande COPY TO (nous verrons d'ailleurs qu'il suffit de copier seulement la structure du fichier et de modifier cette copie). Ensuite, il vous faudra transférer les données de l'ancien fichier dans le nouveau, à l'aide de la commande APPEND FROM.

Nous allons tout d'abord étudier pour elles-mêmes ces nouvelles commandes que sont APPEND FROM et MODIFY STRUCTURE. Auparavant, nous allons voir comment recopier la seule structure d'un fichier à l'aide de COPY STRUCTURE TO... Munis de ces nouveaux outils, nous verrons comment effectuer chacune des cinq modifications évoquées ci-dessus.

1. UN CAS PARTICULIER DE RECOPIE : COPY STRUCTURE TO...

Nous avons vu que la commande CREATE peut nous permettre de créer la structure d'un fichier sans pour autant y placer des renseignements (il suffit de répondre N à la question concernant la saisie). La commande :

COPY STRUCTURE TO (*)

va vous permettre de créer rapidement une nouvelle structure, dans le cas où elle est identique à celle d'un fichier existant. Voici un exemple de recopie de la structure de REPERT dans REPERT2.

```
use repert
. copy structure to repert2
. use repert2
. list
. list structure
STRUCTURE DU FICHIER                : A:REPERT2 .DBF
NOMBRE D'ENREGISTREMENTS            : 00000
DATE DE LA DERNIERE MISE A JOUR : 21/06/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP    NOM          TYP   DIM    DECIMALE(S)
001      NOM           C    010
002      PRENOM        C    008
003      CODEPOS       C    005
004      VILLE         C    008
005      TELEPHONE     C    011
006      TAILLE        N    004    002

** TOTAL **              00047
```

écran 13.1 : copie de la structure d'un fichier

(*) En anglais : copier la structure dans.

Comme nous le confirment les commandes LIST et LIST STRUCTURE, le fichier REPERT2 possède la même structure que REPERT, mais il ne comporte aucun enregistrement.

Remarque :

Cette commande peut ne copier que certains champs d'une structure. Par exemple, dans notre cas :

COPY STRUCTURE TO REPERT2 FIELDS NOM, TELE

créerait une nouvelle structure ne comportant que les champs NOM et TELE.

2. POUR AJOUTER "AUTOMATIQUEMENT" DES ENREGISTREMENTS À UN FICHIER : APPEND FROM

La commande APPEND permet d'ajouter "manuellement" des enregistrements à un fichier. Mais Dbase vous permet également d'ajouter "automatiquement" à un fichier des enregistrements provenant d'un autre fichier grâce à la commande :

APPEND FROM (*)

La situation la plus naturelle correspond au cas où les deux fichiers ont la même structure. Mais nous allons voir que la commande APPEND FROM fonctionne également avec des fichiers de structure différente, pour peu qu'ils aient certains champs en commun. C'est d'ailleurs cette dernière possibilité que nous utiliserons largement dans le cadre de la "modification de structure".

Voyons quelques "situations types"

2.1. Fichiers de même structure

Actuellement REPERT2 possède la même structure que REPERT, mais il est vide. Créons quelques enregistrements par APPEND, de manière à ce qu'il se présente ainsi :

(*) En anglais : ajouter à partir de, depuis.

```
. list
00001  Fournier   Gerard   14200 SISTERON 92 13 45 21  1.92
00002  Lethrosne  Michel   77780 CHIROLS  75 67 28 30  1.77
```

Avec la commande APPEND FROM, nous pouvons demander à Dbase d'ajouter les enregistrements de REPERT2 au fichier REPERT.

```
. use repert
. list
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.57
00003  Taillefert Claude   45510 SULLY    77 89 63 10  1.80
00004  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
00005  Durand     Albert   44444 PARIS    56 55 55 44  1.84
00006  Duval      Alfred   55555 SULLY    44 44 44 44  1.73
. append from repert2
. list
00001  Thomas     Michel   58000 NEVERS   86 48 23 37  1.75
00002  Mitenne    Laurence 83600 FREJUS   89 55 66 89  1.57
00003  Taillefert Claude   45510 SULLY    77 89 63 10  1.80
00004  Grillon    Jules    75006 PARIS    42 48 15 30  1.68
00005  Durand     Albert   44444 PARIS    56 55 55 44  1.84
00006  Duval      Alfred   55555 SULLY    44 44 44 44  1.73
00007  Fournier   Gerard   14200 SISTERON 92 13 45 21  1.92
00008  Lethrosne  Michel   77780 CHIROLS  75 67 28 30  1.77
```

écran 13.2 : APPEND FROM dans le cas de fichiers de même structure

2.2. Fichiers de structure différente

Créons un nouveau fichier REPERT3 ne comportant que trois champs :

- deux communs avec REPERT : NOM et VILLE (avec les mêmes caractéristiques de type et de longueur).
- un nouveau : AGE de type numérique et de longueur 2.

La phase de définition de structure apparaît dans l'écran 13.3. Par contre, la phase de saisie (plein écran) est ''matérialisée'' par des pointillés. Il vous sera facile de la reconstituer à l'aide de la liste du fichier, après création.

```
. create repert3
DONNEZ LA STRUCTURE DE L'ENREGISTREMENT SELON LE FORMAT :
CHAMP     NOM,TYPE,DIMENSION,DECIMALE(S)
 001      nom,c,10
 002      ville,c,8
 003      age,n,2
 004
*** VOULEZ-COMMENCER LA SAISIE (Y/N) ? Y
            •
            •
            •
            •
            •
. list
*** AUCUNE BASE N'EST UTILISEE, DONNEZ SON NOM : repert3
00001  Vidal       Marcel     38
00002  Benoist     Magali     25
```

écran 13.3 : création d'un fichier REPERT3

Utilisons maintenant APPEND FROM pour ajouter à REPERT les enregistrements de REPERT3 :

```
. use repert
. append from repert3
. list
00001  Thomas      Michel    58000 NEVERS   86 48 23 37  1.75
00002  Mitenne     Laurence  83600 FREJUS   89 55 66 89  1.57
00003  Taillefert  Claude    45510 SULLY    77 89 63 10  1.80
00004  Grillon     Jules     75006 PARIS    42 48 15 30  1.68
00005  Durand      Albert    44444 PARIS    56 55 55 44  1.84
00006  Duval       Alfred    55555 SULLY    44 44 44 44  1.73
00007  Fournier    Gerard    14200 SISTERON 92 13 45 21  1.92
00008  Lethrosne   Michel    77780 CHIROLS  75 67 28 30  1.77
00009  Vidal                       Marcel                0.00
00010  Benoist                     Magali                0.00
```

écran 13.4 : APPEND FROM dans le cas de fichiers de structure différente

Voyez comme Dbase a créé 3 nouveaux enregistrements, en prenant dans REPERT3 les informations des champs communs aux deux fichiers.

Remarque :

APPEND FROM considère que deux champs sont communs aux deux fichiers, à partir du moment où ils ont le *même nom*. Toutefois, lorsqu'ils sont de type différent, des conversions automatiques sont

réalisées. Lorsqu'elles ont lieu dans le sens numérique → caractère, aucun problème particulier ne se pose. Par contre, dans le sens caractère → numérique, il est évident que certaines choses n'ont pas de signification : quel nombre associer à la chaîne Thomas ? Dans ce cas, Dbase se contente de traduire cela par 0 (zéro) sans autre forme de procès !

3. POUR MODIFIER LA STRUCTURE D'UN FICHIER : MODIFY STRUCTURE

La commande :

MODIFY STRUCTURE

va vous offrir les possibilités suivantes :

— ajouter un nouveau champ
— supprimer un champ
— modifier le nom, la taille, le type d'un champ.

Nous vous rappelons qu'elle détruit les données du fichier auquel elle s'applique (le fichier courant), de sorte qu'en général on travaille sur une copie de structure.

Pour expérimenter cette commande, nous vous proposons de refaire une copie de la structure de REPERT dans REPERT2 puis de demander la modification de la structure de REPERT2

```
. use repert
. copy structure to repert2
. use repert2
. modify structure
LA COMMANDE "MODIFY" DETRUIRA TOUTES VOS DONNEES (Y/N) ? Y
```

écran 13.5 : pour modifier la structure d'un fichier : MODIFY STRUCTURE

Dbase vous affiche alors, en plein écran, la structure complète, sous la forme suivante :

	NOM	TYP	DIM	DEC
CHAMP 01	NOM	C	010	000
CHAMP 02	PRENOM	C	008	000
CHAMP 03	CODEP	C	005	000
CHAMP 04	VILLE	C	008	000
CHAMP 05	TELE	C	011	000
CHAMP 06	TAILLE	N	004	002
CHAMP 07				
CHAMP 08				
CHAMP 09				
CHAMP 10				
CHAMP 11				
CHAMP 12				
CHAMP 13				
CHAMP 14				
CHAMP 15				
CHAMP 16				
CHAMP 17				
CHAMP 18				
CHAMP 19				
CHAMP 20				
CHAMP 21				
CHAMP 22				

écran 13.6 : le mode modification de structure

Comme en mode EDIT, vous pouvez effectuer des modifications, à l'intérieur des différents champs à l'aide des touches :

Changement d'enregistrement	Suivant	Alt/C	(*)
	précédent	Alt/R	(*)
Terminer les corrections	Normalement	Alt/W	(*)
	En ignorant les corrections de l'enregistrement courant	Alt/Q	(*)

(*) Sur CPC, Alt sera remplacé par Ctrl.

tableau 13.7 : les touches de modification d'un champ d'une structure

D'autres touches vous permettent de supprimer un champ ou de préparer une ligne blanche pour en définir un nouveau :

Supprimer le champ dans lequel se trouve le curseur	Alt/T	(*)
Insérer une ligne blanche au-dessus du champ où se trouve le curseur	Alt/N	(*)
Terminer la modification	Alt/W	(*)

tableau 13.8 : les touches d'insertion ou de suppression d'un champ d'une structure.

▷ **Entraînez-vous**

- Poursuivez la manipulation que nous avons décrite, de manière à ce que le fichier REPERT2 possède la structure suivante (vérifiez par LIST STRUCTURE).

NOM	C	15
PRENOM	C	12
CODEPOS	N	5
LIEU	C	8
TELEPHONE	C	11
TAILLE	C	4

- Modifiez à nouveau la structure, de façon à supprimer les champs PRENOM et TAILLE. Vérifiez.
- A partir de cette dernière structure, revenez à la structure d'origine (c'est-à-dire celle de REPERT). Vérifiez.

4. POUR AJOUTER UN CHAMP À UN FICHIER EXISTANT

Nous disposons de tous les outils nécessaires. A titre d'exemple, supposons que nous souhaitions ajouter à notre fichier STOCK un champ de type numérique, destiné à contenir l'information : ''stock minimum de renouvellement''. Bien entendu, nous souhaitons pouvoir continuer à disposer des informations y figurant déjà. Nous procédons ainsi :

```
. use stock
. copy structure to stockbis
. use stockbis
. modify structure
LA COMMANDE "MODIFY" DETRUIRA TOUTES VOS DONNEES (Y/N) ? Y
          •
          •
          •
          •
          •
. list structure
STRUCTURE DU FICHIER                : A:STOCKBIS.DBF
NOMBRE D'ENREGISTREMENTS            : 00000
DATE DE LA DERNIERE MISE A JOUR : 21/06/86
BASE DE DONNEES PRIMAIRE EN COURS D'UTILISATION
CHAMP    NOM          TYP    DIM     DECIMALE(S)
001      PRODUIT       C     020
002      CAT           C     001
003      REF           C     003
004      PRIX          N     008     002
005      MINIMUM       N     005
006      QTE           N     005

** TOTAL **                 00043
. append from stock
00006 ENREGISTREMENT(S)  AJOUTE(S)
. list
00001  Cafetière 12 T        A 432    137.47      0      32
00002  Four à micro-ondes    E 248   1875.25      0       7
00003  Grille pain           A 521    158.75      0      23
00004  Four à raclette 6 P   A 427    133.72      0      15
00005  Friteuse 1 kg         B 433    349.25      0      21
00006  Table de cuisson      E 647   2450.00      0       6
```

écran 13.9 : pour ajouter un nouveau champ (*)

Notez bien que pour l'instant, nous nous trouvons en présence de deux fichiers : l'ancien STOCK et le nouveau STOCKBIS. Autrement dit, nous n'avons pas réellement modifié la structure de STOCK ; nous avons simplement créé un nouveau fichier STOCKBIS ayant la structure souhaitée et dans lequel nous avons reporté les informations de STOCK.

Si vous souhaitez ne plus avoir à faire qu'à un seul fichier nommé toujours STOCK, il vous faudra :

— effacer l'ancien fichier par :

- DELETE FILE STOCK

— donner à l'actuel fichier STOCKBIS le nom STOCK par :

- USE
- RENAME STOCKBIS TO STOCK

(*) La phase de modification de structure en plein écran est matérialisée par des points.

Notez que cette (nouvelle) commande RENAME ne peut pas travailler sur le fichier courant. Il est donc nécessaire de le ''fermer'' par USE.

Remarque :

Le nouveau champ MINIMUM que nous venons de créer ne comporte naturellement aucune information(*). Il vous sufffira de la saisir par EDIT, EDIT FIELDS, BROWSE ou BROWSE FIELDS.

5. POUR SUPPRIMER UN CHAMP D'UN FICHIER

Le procédé est voisin du précédent. Nous vous proposons de l'essayer par vous-même.

▷ **Entraînez-vous**

- Créez, à partir du fichier REPERT, un fichier nommé REPERBIS comportant les mêmes champs, à l'exception du champ TAILLE.

6. POUR MODIFIER LA TAILLE D'UN CHAMP

Là encore, le déroulement des opérations ressemble à ce que nous venons de voir. Il faut simplement savoir qu'en cas de diminution de taille, certaines informations risquent de se trouver ''tronquées'' (c'est le cas dans l'''entraînez-vous'' que nous vous proposons)

▷ **Entraînez-vous**

- Créez, à partir de REPERT un fichier nommé REPERBIS ayant la structure suivante :

NOM	C	8	
PRENOM	C	6	
CODEPOS	N	5	
VILLE	C	6	
TAILLE	N	6	2

  Vérifiez le résultat obtenu.
- Même question en donnant la description suivante au champ taille :

TAILLE	N	5	1

- Même question avec :

TAILLE	N	3	2

(*) Ce qui, en numérique, se traduit par 0.

7. POUR MODIFIER LE NOM D'UN CHAMP

Supposez que nous souhaitions créer un fichier STOCKBIS obtenu à partir de STOCK en remplaçant le nom de champ CAT par CATEGORIE.

Si nous procédons comme dans les paragraphes 4 à 6 (copie de la structure, modification du nom de champ et APPEND FROM), nous obtenons un fichier de structure convenable mais dans lequel le champ CATEGORIE est vide. En effet, la commande APPEND FROM ne fonctionne que sur les champs communs aux deux fichiers (c'est-à-dire les champs de même nom).

Si l'on souhaite que ce nouveau champ soit convenablement renseigné, il est nécessaire de procéder en plusieurs étapes :

1. Créer, à partir de STOCK, un nouveau fichier, par exemple STOCTER comportant les mêmes champs que STOCK plus un nouveau champ nommé CATEGORIE, de même type et de même longueur que CAT (le champ CAT est donc toujours présent).
2. Reporter, dans STOCKER, toutes les valeurs du champ CAT dans ce nouveau champ CATEGORIE, à l'aide de :
   - USE STOCKTER
   - REPLACE ALL CATEGORIE WITH CAT
3. Créer, à partir de STOCKTER, le fichier STOCKBIS, obtenu par suppression du champ CAT, devenu (maintenant) inutile.

▷ **Entraînez-vous**

- Réalisez la procédure proposée ci-dessus, en vérifiant soigneusement chacune des étapes.
- Faites de même avec REPERT pour créer un fichier REPERTER, de même structure que REPERT, mais dans lequel le champ VILLE est devenu LIEU et le champ CODEP est devenu CPOSTAL.

XIV Premières notions de programmation

Au cours des précédents chapitres, nous avons fait connaissance avec les principales commandes de Dbase et nous les avons utilisées en mode ''conventionnel''. Autrement dit, nous frappions ces commandes au clavier et Dbase les exécutait sur le champ.

Or, comme nous l'avons laissé entrevoir dans le chapitre 1, Dbase possède des possibilités de programmation. Qu'est-ce programmer ? Nous pouvons dire succinctement que cela consiste à ''enregistrer'' un ensemble de commandes qui porte alors le nom de ''programme''. Ultérieurement, ce programme peut être exécuté autant de fois que souhaité, évitant ainsi d'avoir à fournir à nouveau la liste des commandes qui le constituent.

La programmation apparaît donc comme un moyen d'automatiser certaines tâches. Mais elle ne se réduit pas à cela. En effet, en Dbase, certaines commandes (nouvelles) nous permettront de réaliser ce que l'on nomme des structures. Plus précisément, nous trouverons :

— *la structure de choix :* le programme pourra choisir entre plusieurs possibilités (correspondant chacune à des commandes différentes). Le ''comportement'' du programme pourra ainsi être différent d'une exécution à l'autre. Il semblera ''*prendre des décisions*''. En réalité, ces décisions seront strictement conditionnées par des informations provenant, soit des fichiers traités, soit de l'utilisateur lui-même

— *la structure de répétition :* les mêmes commandes pourront être répétées plusieurs fois. Par exemple, un certain traitement défini pour un enregistrement pourra être répété pour tous les enregistrements du fichier.

Dans ce chapitre, nous allons commencer par vous montrer comment fabriquer un programme et l'utiliser. Pour ce faire, nous choisirons un ensemble de commandes connues que nous nous contenterons donc d'automatiser. Nous aborderons ensuite l'importante notion de variable et les commandes permettant à l'utilisateur de communiquer des informations au programme. Nous disposerons alors des bases nécessaires pour aborder dans les chapitres suivants les commandes de structuration.

1. CRÉER ET EXÉCUTER UN PROGRAMME

Supposez que nous ayons souvent besoin d'obtenir la liste des articles de catégorie A et E de notre fichier STOCK. Pour ce faire, nous sommes souvent amenés à exécuter ces commandes :

```
. use stock
. list for cat="A"
00001  Cafetière 12 T        A 432    137.47     32
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
. list for cat="E"
00002  Four à micro-ondes    E 248   1875.25      7
00006  Table de cuisson      E 647   2450.00      6
```

écran 14.1

1.1. Pour créer un programme : MODIFY COMMAND

La commande :

MODIFY COMMAND

nous permet d'enregistrer une suite de commandes dans un fichier. (Ce dernier, nommé également ''fichier de commandes'' possèdera l'extension CMD).

Comme MODIFY STRUCTURE, elle travaille en mode plein écran et elle utilise les mêmes touches (voir tableaux pages 121 et 122). Il faut

simplement considérer que, lorsque vous créez votre programme, votre écran est vide (tandis qu'avec MODIFY STRUCTURE, il affichait la structure existante). D'autre part, ici, chaque ligne correspond à une commande, tandis que dans MODIFY STRUCTURE, chaque ligne correspondait à la description d'un champ.

▷ **Entraînez-vous**

- A l'aide de MODIFY COMMAND, enregistrez dans un fichier nommé LISTOCK la suite des trois commandes :

  USE STOCK

  LIST FOR CAT = "A"

  LIST FOR CAT = "E"

(Notez bien qu'il ne faut pas taper de point en début de ligne – n'oubliez pas de terminer par ALT/W)

1.2. Pour exécuter un programme : DO

Pour demander à Dbase d'exécuter les commandes enregistrées dans un fichier, il nous suffit d'utiliser la commande :

DO

en spécifiant le nom du fichier en question

```
. do listock
00001  Cafetière 12 T        A 432    137.47     32
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
00002  Four à micro-ondes    E 248   1875.25      7
00006  Table de cuisson      E 647   2450.00      6
```

écran 14.2 : pour exécuter un programme : DO

▷ **Entraînez-vous**

- Vérifiez que vous pouvez exécuter ce programme autant de fois que vous le voulez, y compris après avoir quitté Dbase (et y être revenu)

Remarque :

Dbase ne reproduit pas à l'écran les commandes qu'il exécute. Voyez la différence entre l'écran 12.1 (où nous avions frappé chaque commande au clavier) et l'écran 12.2 (où les commandes sont directement "prélevées" dans le fichier LISTOCK).

2. POUR "REVOIR" UN PROGRAMME

Il n'existe pas de commande permettant de "lister" le contenu d'un "fichier de commandes" (ou programme). Par contre, la commande MODIFY COMMAND permet aussi bien la création d'un fichier de commande que sa modification. Dans ce dernier cas, le contenu du fichier apparaît alors à l'écran.

Ainsi, par exemple, pour revoir le contenu de LISTOCK, il vous suffira d'exécuter :

- MODIFY COMMAND LISTOCK

puis de "quitter" par Alt/W sans effectuer de modifications.

Remarque

Un fichier de commande est un vrai fichier texte, au sens du système CP/M (bien que son extension ne soit pas TXT). Dans ces conditions, il vous est possible d'en obtenir une liste, lorsque vous êtes "sous système", par la commande CP/M : **TYPE.** Dans ce cas, il faudra bien préciser l'extension CMD (laquelle était "prise par défaut" par la commande MODIFY COMMAND de Dbase). Ainsi, pour lister LISTOCK, nous exécuterons :

A > TYPE LISTOCK.CMD

3. EN CAS D'ERREUR DE PROGRAMME

Lorsque vous "saisissez" vos commandes sous MODIFY COMMAND, Dbase se contente d'enregistrer le "texte" que vous lui fournissez. Il n'effectue aucune vérification à ce niveau, de sorte que vous pouvez très bien y introduire des commandes erronées.

Dans ces conditions, vous voyez que ce n'est que lors de *l'exécution* du programme que Dbase sera en mesure de découvrir une éventuelle erreur. Il la signalera alors comme il l'aurait fait si vous aviez frappé la même commande au clavier. De même, il vous demandera si vous souhaitez apporter des corrections ; nous vous conseillons d'éviter cette possibilité, au profit de la correction du programe lui-même (par MODIFY COMMAND)

▷ **Faites-vous la main**

- Remplacez (par MODIFY COMMAND) la première ligne de LISTOCK par :

  US STOCK

- Cherchez à exécuter LISTOCK et voyez ce qui se produit.
- Essayez d'autres "erreurs" de votre choix.
- Redonnez son état initial à LISTOCK.
- Réalisez et utilisez un programme formé des commandes suivantes :

  USE REPERT

  INDEX ON NOM TO TEMPO

  LIST

  DELETE FILE TEMPO.INDEX

Voyez comme son comportement n'est pas satisfaisant. Effectuez la modification appropriée.

4. LA NOTION DE VARIABLE

Actuellement, vous pouvez manipuler l'information contenue dans des champs, la modifier ou afficher à l'écran des résultats de traitement. Mais, dans certaines circonstances, vous aurez besoin de conserver une information (autrement qu'en la plaçant dans un champ, donc dans un fichier), ceci afin de pouvoir l'exploiter plus tard. Cela sera possible en employant ce que l'on nomme une "variable".

Cette nouvelle notion ne prendra son véritable intérêt qu'au sein d'un programme. Nous allons toutefois commencer par vous la présenter en "mode conversationnel"(*)

4.1. Une façon de mettre une valeur dans une variable : STORE

Essayez ces manipulations

(*) Nous appelons "mode conversationnel", le mode qui consiste à frapper les commandes au clavier. Il s'oppose au "mode programme", dans lequel les commandes sont prélevées par Dbase dans un "fichier de commandes". Le mode conversationnel est aussi appelé "mode direct".

```
. store 5 to a
  5
. store 12 to b
  12
. ? a
          5
. ? b
         12
. ? a+b
          17
```

écran 14.3 : pour conserver des valeurs dans des variables : STORE.

La commande :

STORE 5 TO A

signifie : ranger la valeur 5 dans un emplacement de la mémoire que l'on conviendra de nommer A.

Notez qu'après avoir exécuté cette instruction, Dbase nous affiche la valeur que la commande STORE l'a amené à ranger en mémoire.

Rappelons que ? permet d'afficher la valeur d'une ''expression''. Jusqu'ici, nous avions simplement mentionné qu'une expression pouvait comporter des noms de variable, mais sans en voir d'exemples. Ici, nous rencontrons deux expressions réduites à un simple nom de variable : A, puis B (un peu comme nous avions rencontré des expressions réduites à un nom de champ). Nous trouvons ensuite une expression (A + B) exprimant la somme des deux variables A et B.

Remarque

les ''noms'' de variable sont constitués suivant les mêmes règles que les noms de champ.

4.2. La valeur d'une variable peut évoluer

Essayez ces manipulations :

```
. ? a
          5
. store 10 to a
  10
. ? a
         10
. ? a+b
          22
```

écran 14.4 : quand la valeur d'une variable varie !

Voyez comme la commande :

STORE 10 TO A

a rangé la valeur 10 dans A, supprimant du même coup la valeur (5) qui s'y trouvait auparavant.

Essayez maintenant :

```
. ? a
          10
. store a+1 to a
  11
. ? a
          11
```

écran 14.5 : pour augmenter de 1 la valeur d'une variable

La commande :

STORE A + 1 TO A

demande de ranger dans A la valeur de l'expression A + 1. Au bout du compte, cela revient à augmenter de 1 la valeur de A, quelle que soit cette valeur.

Une telle instruction se révèlera très précieuse, dans un programme, pour ''compter'' quelque chose, en particulier des ''tours de boucle''.

▷ **Entraînez-vous**

- Vérifiez que les variables A et B contiennent bien les valeurs 11 et 12 (sinon modifiez-les).
- Que font ces commandes ; vérifiez-le en les exécutant
  - STORE A + B TO SOM
  - STORE A * B TO PROD
  - STORE 2 * SOM TO SOM
  - ? ''le double de'', A, ''est'', 2 * A
  - ? A, ''et'', B, ''ont pour somme'', A + B
  - ? A, ''et'', B, ''ont pour somme'', S.
- Placez la valeur 1,186 dans une variable nommée TAUX.

- Exécutez ces commandes :
  - USE REPERT
  - LIST REF, PRIX * TAUX.
- Remplacez la valeur de TAUX par 1.333 et exécutez à nouveau les commandes ci-dessus.
- Voyez ce qui se produit lorsque vous cherchez à utiliser une variable inexistante, comme dans :
  - STORE C TO A.

5. LE TYPE D'UNE VARIABLE

Nous avons introduit la notion de variable en y rangeant des informations de type "numérique" (des nombres). En fait, comme dans les cas des "champs", Dbase nous permet également d'y placer des informations de type "caractère"(*). Voyez ces exemples :

```
. store "bonjour " to mes
bonjour
. ? mes
bonjour
. ? $(mes,1,3)
bon
. store "monsieur" to pers
monsieur
. ? mes+pers
bonjour monsieur
. use repert
. list mes+nom
00001  bonjour Thomas
00002  bonjour Mitenne
00003  bonjour Loiseau
00004  bonjour Meunier
00005  bonjour Taillefert
00006  bonjour Grillon
. store "hello " to mes
hello
. list mes+nom
00001  hello Thomas
00002  hello Mitenne
00003  hello Loiseau
00004  hello Meunier
00005  hello Taillefert
00006  hello Grillon
```

écran 14.6 : exemples d'utilisation de variables de type caractère.

(*) Dbase vous permet également d'y placer des informations de type "logique" (dont la valeur est soit vrai, soit faux).

Remarque :

Contrairement à ce qui se produit dans la majorité des langages de programmation, le type des informations placées dans une variable peut évoluer au fil du temps. En voici un exemple :

```
. store 10 to a
  10
. ? a
          10
. store "bonjour" to a
bonjour
. ? a
bonjour
```

écran 14.7 : quand une variable change de type

6. POUR CONNAÎTRE TOUTES LES VARIABLES UTILISÉES : DISPLAY MEMORY

Lorsque vous manipulez beaucoup de variables, il peut vous arriver de ne plus savoir exactement où vous en êtes. Dbase peut venir à votre secours à l'aide de la commande :

DISPLAY MEMORY

```
. display memory
A            (C)  bonjour
B            (N)    12
SUM          (N)    46
PROD         (N)    132
MES          (C)  hello
PERS         (C)  monsieur

** TOTAL **      06 VARIABLES        00045 OCTETS
```

écran 14.8 : pour connaître les variables utilisées : DISPLAY MEMORY

Cette commande vous fournit, pour chaque variable

- son nom
- le type de l'information qui s'y trouve à cet instant (C pour caractère, N pour numérique)
- son contenu.

En outre, elle vous indique le nombre total de variables utilisées et la place (exprimée en ''octets'') qu'elles occupent en mémoire.

Remarque :

Sachez que vous ne pouvez utiliser plus de **64 variables** différentes et que l'espace mémoire qui leur est attribué par Dbase est limité à **1536 octets**. Pour vous aider à ''mesurer'' ce que cela représente, sachez que :

- une variable contenant un nombre occupe 7 octets
- une variable contenant une chaîne occupe un octet par caractère plus un.

7. POUR SUPPRIMER DES VARIABLES INUTILES : RELEASE

Tout d'abord, il faut bien voir que les emplacements réservés aux variables sont situés dans la ''mémoire centrale'' de votre ordinateur et non sur disquette. Cela signifie donc que vos variables disparaissent à la mise hors tension, et même lorsque vous quittez Dbase.

Malgré leur caractère fugitif, il peut vous arriver d'être encombré par des variables devenues inutiles. La commande :

RELEASE

vous permet de détruire tout ou partie de vos variables.

```
. display memory
A            (C)  bonjour
B            (N)    12
SUM          (N)    46
PROD         (N)    132
MES          (C)  hello
PERS         (C)  monsieur

** TOTAL **      06 VARIABLES        00045 OCTETS
. release sum
. display memory
A            (C)  bonjour
B            (N)    12
PROD         (N)    132
MES          (C)  hello
PERS         (C)  monsieur

** TOTAL **      05 VARIABLES        00038 OCTETS
. release all
. display memory

** TOTAL **      00 VARIABLES        00000 OCTETS
```

écran 14.9 : pour détruire des variables : RELEASE

La commande RELEASE SUM détruit la variable nommée SUM. La commande RELEASE ALL détruit *toutes* les variables.

8. POUR CONSERVER DANS UNE VARIABLE LES RÉSULTATS DE COUNT ET SUM

Dans ces deux commandes, vous pouvez spécifier (à l'aide du paramètre TO) le nom d'une variable destinée à recevoir le résultat. En voici des exemples :

```
. use stock
. count for cat="A" to nba
** COMPTEUR = 00003
. sum prix*qte for cat="A" to vala
 10056.09
. ? "Les", nba, "articles de catégorie A valent", vala
Les           3 articles de catégorie A valent    10056.09
```

écran 14.10 : pour conserver les résultats de COUNT et SUM

9. UN PROGRAMME UTILISANT DES VARIABLES

Nous vous avons familiarisé avec la notion de variable en nous servant des commandes en mode conversationnel. Voici maintenant un exemple de programme utilisant des variables : il s'agit tout simplement des commandes du paragraphe 8 que nous enregistrons (avec MODIFY COMMAND) dans un fichier nommé VALSTOCK, à savoir :

```
use stock
count for cat="A" to nba
sum prix*qte for cat="A" to vala
? "Les", nba, "articles de catégorie A valent", vala
```

écran 14.11 : programme de calcul de la valeur du stock des articles de catérogie A

Exécutons ce programme :

```
. do valstock
** COMPTEUR = 00003
 10056.09
Les            3 articles de catégorie A valent     10056.09
```

écran 14.12 : exécution du programme 14.11

Vous constatez qu'un certain nombre d'informations (nombre d'articles et valeur) apparaissent de manière peu naturelle, alors qu'elles ne choquaient pas en mode conversationnel. Il est effectivement acceptable d'avoir une sorte de ''compte rendu d'exécution'' d'une commande frappée au clavier. Il est généralement plus désagréable de voir un tel compte rendu apparaître dans le cas d'un programme (imaginez qu'il comporte 100 commandes ! ou encore pensez à la perplexité de l'utilisateur néophyte).

En fait, il existe une commande demandant à Dbase de ne plus effectuer de ''compte rendu'' ; il s'agit de :

SET TALK OFF (*)

Incorporons dans notre programme en le faisant se terminer par la commande d'annulation de l'effet précédent :

SET TALK ON

(*) En anglais TALK signifie parler.

Notre programme se présente alors ainsi :

```
set talk off
use stock
count for cat="A" to nba
sum prix*qte for cat="A" to vala
? "Les", nba, "articles de catégorie A valent", vala
set talk on
```

écran 14.13 : programme sans "compte rendus"

En voici un exemple d'exécution :

```
. do valstock
Les            3 articles de catégorie A valent     10056.09
```

écran 14.14 : exécution du programme 14.13

▷ **Entraînez-vous**

(Voir correction en fin de volume).

- (1) Créez un programme affichant la taille moyenne des "Parisiens" du fichier REPERT

10. POUR COMMUNIQUER DES INFORMATIONS À UN PROGRAMME : ACCEPT et INPUT

Le programme de l'écran 14.13 détermine la valeur du stock des articles de catégorie A. Comment obtenir la valeur du stock des articles d'une autre catégorie. Nous pourrions bien sûr :

— modifier le programme précédent en y adaptant la catégorie. Ce serait toutefois une solution peu agréable pour nous et encore moins pour un néophyte simple utilisateur de notre programme.

— écrire autant de programmes différents qu'il y a de catégories différentes. C'est là une solution peu élégante mais acceptable avec trois catégories ; mais s'il y en avait 50 ?

Une bien meilleure solution consiste à réaliser un programme unique auquel on "communique" d'une manière "ad'hoc" la catégorie intéressante.

10.1. Une première solution "assez rustique"

Il existe une première manière de procéder qui ne fait appel qu'aux notions que nous avons déjà rencontrées, à savoir :

- placer dans un premier temps la catégorie cherchée dans une variable, à l'aide d'une commande STORE, en mode direct.
- utiliser un programme dans lequel la catégorie recherchée est, non plus une constante, mais celle figurant dans la variable en question.

Si nous choisissons d'appeler cette variable CATRECH (pour Catégorie Recherchée), le programme pourrait se présenter ainsi :

```
set talk off
use stock
count for cat=catrech to nb
sum prix*qte for cat=catrech to val
? "Les",nb,"articles de catégorie",catrech,"valent",val
set talk on
```

écran 14.15

En voici quelques exemples d'exécution :

```
. store "C" to catrech
C
. do valstock
Les           0 articles de catégorie C valent        0.00
. store "E" to catrech
E
. do valstock
Les           2 articles de catégorie E valent    27826.75
. store "A" to catrech
A
. do valstock
Les           3 articles de catégorie A valent    10056.09
```

écran 14.16 : exemples d'exécution du programme de l'écran 14.15

Néanmoins, vous constatez que l'emploi du nouveau programme VALSTOCK nécessite la connaissance d'au-moins la commande STORE de Dbase. En outre, il faut se souvenir du nom (CATRECH) de la variable destinée à recevoir la catégorie cherchée.

Nous allons voir comment la commande ACCEPT nous fournit une meilleure solution à notre problème.

10.2. Une seconde solution : ACCEPT

La commande :

ACCEPT

permet d'introduire dans une variable une information frappée au clavier. Voici comment nous pouvons l'incorporer dans notre programme :

```
set talk off
accept "quelle catégorie " to catrech
use stock
count for cat=catrech to nb
sum prix*qte for cat=catrech to val
? "Les",nb,"articles de catégorie",catrech,"valent",val
set talk on
```

écran 14.17 : programme de calcul de la valeur du stock des articles de catégorie donnée

En voici quelques exemples :

```
. do valstock
quelle catégorie :B
Les            1 articles de catégorie B valent      7334.25
. do valstock
quelle catégorie :E
Les            2 articles de catégorie E valent     27826.75
. do valstock
quelle catégorie :A
Les            3 articles de catégorie A valent     10056.09
```

écran 14.18 : exemples d'exécution du programme de l'écran 14.17

La commande :

ACCEPT ''quelle catégorie vous intéresse ?'' TO CATCHER

fait les choses suivantes :

— elle affiche à l'écran la chaîne placée après le mot ACCEPT

— puis attend que vous frappiez une information au clavier, et que vous signaliez que vous avez terminé en tapant ''RETURN''

— enfin, elle range l'information ainsi fournie dans la variable mentionnée après TO, ici CATCHER.

10.3. ACCEPT ou INPUT ?

La commande ACCEPT crée automatiquement une variable de type caractère, même si l'information que vous lui fournissez est numérique. Si vous souhaitez absolument obtenir une variable numérique, il est nécessaire d'employer une commande voisine :

INPUT

Ainsi, par exemple :

INPUT ''combien en voulez-vous ?'' TO NOMB

créera effectivement une variable NOMB de type numérique.

Par contre, une telle instruction ne pourra plus accepter n'importe quelle réponse (voyez le prochain ''Entraînez-vous'').

▷ **Entraînez-vous**

(Les points comportant un numéro sont corrigés en fin de volume).

- Réalisez ce petit programme pour examiner comment INPUT ''interprète'' des réponses ''plus ou moins'' numériques :

  INPUT ''combien'' TO N

  ? ''j'ai compris :'', N

- Essayez-le à diverses reprises, en répondant :

  345

  43.28

  3M48

  4XC5B

BONJOUR

(Vous pouvez examiner, dans chaque cas, le type de N par DISPLAY MEMORY).

- (2) Réalisez un programme comptant, dans REPERT, le nombre d'habitants d'une ville donnée.
- (3) Réalisez un programme calculant, à partir de REPERT, la taille moyenne des individus de prénom donné.
- (4) Réalisez un programme marquant pour effacement tous les enregistrements de REPERT correspondants à une ville donnée.
- (5) Réalisez un programme augmentant d'un pourcentage donné le prix de tous les articles d'une catégorie donnée.

XV Pour qu'un programme prenne des décisions : les "structures" de choix : IF... et DO CASE...

Les programmes du précédent chapitre voyaient leurs commandes s'exécuter "séquentiellement" dans l'ordre où elles avaient été enregistrées dans le fichier. Nous allons maintenant apprendre comment, au sein d'un programme, faire choisir Dbase entre plusieurs possibilités. Pour ce faire, nous disposerons de deux structures de choix :

— le choix entre deux possibilités ; il sera réalisé par les commandes IF, ELSE et ENDIF.
— le choix entre un nombre quelconque de possibilités : il sera réalisé par les commandes DO CASE, CASE et ENDCASE.

1. LA "STRUCTURE" DE CHOIX : IF... ELSE... ENDIF

Commençons par un exemple simple (nommé CHOIX) formé de quelques commandes qui, dans la pratique, ne mériteraient pas vraiment de faire l'objet d'un programme.

```
set talk off
input "donnez un nombre" to n
if n>100
  ? "votre nombre est assez grand"
  ? "vous avez gagné"
else
  ? "votre nombre est trop petit"
  ? "vous avez perdu"
endif
? "au revoir et merci"
set talk on
```

écran 15.1 : la structure de choix

La première ligne attend que l'utilisateur fournisse une valeur et la range dans la variable nommée N. Les sept lignes suivantes expriment un choix. Nous pouvons les shématiser ainsi :

IF N > 100

commandes exécutées lorsque la condition (N > 100) est vraie

ELSE

commandes exécutées lorsque la condition (N > 100) est fausse

ENDIF

Lorsque Dbase rencontre la commande IF, il examine la condition qui suit ce mot. Si cette condition est vraie, il exécute les commandes situées entre IF et ELSE. Dans le cas contraire, il exécute les commandes situées entre ELSE et ENDIF. Ensuite de quoi, que la dite condition soit vraie ou fausse, Dbase passe à la commande suivant ENDIF.

Confirmons tout cela en exécutant plusieurs fois notre programme :

```
. do choix
donnez un nombre:75
votre nombre est trop petit
vous avez perdu
au revoir et merci

. do choix
donnez un nombre:128
votre nombre est assez grand
vous avez gagné
au revoir et merci

. do choix
donnez un nombre:100
votre nombre est trop petit
vous avez perdu
au revoir et merci
```

Ecran 15.2 : exemples d'exécution du programme de l'écran 15.1

Remarques :

Vous pouvez placer autant de commandes que vous le souhaitez dans chaque partie du choix. Pour qu'une telle structure ressorte clairement dans votre programme, nous vous conseillons d'écrire en retrait les commandes constituant chacune des parties du choix (comme nous l'avons fait dans notre exemple).

▷ **Entraînez-vous**

- Assurez-vous de bien saisir la différence entre ces deux programmes (au besoin, vérifiez vos hypothèses en les essayant).

```
INPUT "nombre ?" TO N
IF N > 10
  ? "gagné"
ELSE
  ? "perdu"
  ? "au revoir"
ENDIF
```

```
INPUT "nombre ?" TO N
IF N > 10
  ? "gagné"
ELSE
  ? "perdu"
ENDIF
? "au revoir"
```

- Voyez comment réagit Dbase lorsque vous cherchez à exécuter des programmes erronés tels que :

```
INPUT "nombre ?" TO N
IF N > 10
  ? "gagné"
ENDIF
  ? "perdu"
ELSE
```

```
INPUT "nombre ?" TO N
IF N > 10
  ? "gagné"
ELSE
  ? "perdu"
```

2. UN PROGRAMME À DOUBLE USAGE

Nous vous proposons un autre exemple employant une structure de choix. Il s'agit d'un programme nommé INTSTOCK qui, suivant la demande de l'utilisateur, peut :

— soit déterminer le nombre d'articles d'une catégorie donnée

— soit déterminer la valeur du stock d'articles d'une catégorie donnée.

```
set talk off
use stock
accept "categorie choisie ?" to catcher
? "choisissez entre :"
? " 1 - nombre d'articles de cette catégorie"
? " 2 - valeur du stock d'articles de cette catégorie"
input  "votre choix ? " to choix
if choix = 1
   count for cat = catcher to nart
   ? "Il y a ",nart,"articles de catégorie",catcher
else
   sum prix*qte for cat = catcher to val
   ? "valeur des articles de catégorie",catcher,":",val
endif
? "au revoir"
set talk off
```

écran 15.3 : un programme à double usage

Les trois premières lignes vous sont certainement assez familières. Les trois lignes suivantes affichent trois lignes de texte qui proposent un choix à l'utilisateur. La ligne suivante attend la réponse et en place la valeur (numérique) dans la variable nommée CHOIX.

Les sept lignes suivantes expriment un choix basé sur la condition :

CHOIX = 1

Exécutons plusieurs fois ce programme :

```
. do intstock
categorie choisie ?:A
choisissez entre :
 1 - nombre d'articles de cette catégorie
 2 - valeur du stock d'articles de cette catégorie
votre choix ? :1
Il y a            3 articles de catégorie A
au revoir

. do intstock
categorie choisie ?:E
choisissez entre :
 1 - nombre d'articles de cette catégorie
 2 - valeur du stock d'articles de cette catégorie
votre choix ? :2
valeur des articles de catégorie E :     27826.75
au revoir
```

écran 15.4 : exemples d'exécution du programme de l'écran 15.3.

Remarque :

Le programme propose à l'utilisateur de répondre l'une des deux valeurs 1 ou 2. En fait, compte tenu de ce que l'instruction IF est basée sur la condition CHOIX = 1, toute valeur autre que 1 (et donc pas nécessairement 2) provoque l'exécution de la seconde partie de la structure de choix. Dans un programme réel, on s'arrangera pour refuser toute réponse autre que 1 ou 2 (nous verrons comment procéder dans le chapître suivant).

3. UN PROGRAMME DE RECHERCHE

3.1. Un programme un peu "rustique"

Supposez que nous ayons réalisé le petit programme de recherche ci-dessous. Il permet (en principe) de retrouver le nom d'une personne correspondant à un numéro de téléphone donné.

Vous constatez que tout va très bien lorsque le numéro recherché figure effectivement dans le fichier.

```
use repert
accept "telephone (8chiffres)" to telerech
locate for tele = telerech
? "nom recherché", nom, prenom
```

```
. do cherche
telephone (8chiffres):89 55 66 89
*** ENREGISTREMENT : 00002
nom recherché Mitenne     Laurence

. do cherche
telephone (8chiffres):55 55 55 55
*** FIN DE FICHIER ***
nom recherché Grillon     Jules
```

écran 15.5 : programme (rustique) de recherche d'un abonné de numéro donné

Par contre, dès qu'il n'y figure pas, vous constatez que nous obtenons :

— le message *** FIN DE FICHIER *** ; certes, pour vous qui connaissez Dbase, ce message vous laisse deviner que le numéro n'a pas été trouvé. Mais, pour un néophyte utilisant ce programme !

— un nom ! Il s'agit en fait du nom correspondant au dernier enregistrement du fichier. Ceci provient de ce que, lorsque Dbase atteint la fin du fichier, le pointeur est toujours positionné sur le dernier enregistrement.

La situation n'est donc pas très satisfaisante. Notez qu'elle le serait encore moins si nous avions (comme à l'accoutumé), supprimé les compte rendus. En effet, dans ce cas, nous ne verrions même plus apparaître le message de fin de fichier.

3.2. Pour améliorer la situation

En fait, il faudrait que notre programme puisse choisir entre :

— afficher un message du type ''ce numéro ne figure pas au fichier'' dans le cas où la recherche a été infructueuse

— afficher le nom cherché dans le cas où la recherche a été fructueuse.

Cette fois, la condition régissant le choix serait : fin de fichier atteinte.

Comment exprimer une telle condition en Dbase ? Il existe une fonction, notée :

EOF

EOF est l'abrévation de ''End Of File'' (fin de fichier). Elle prend l'une des deux valeurs : vrai ou faux suivant que la fin du fichier courant a été ou non atteinte.

D'où le nouveau programme assortit de deux exemples d'exécution.

```
set talk off
use repert
accept "telephone (8chiffres)" to telerech
locate for tele = telerech
if eof
    ? "ce numéro ne figure pas au fichier"
  else
    ? "nom recherché", nom, prenom
endif
set talk on
```

```
. do cherche
telephone (8chiffres):89 55 66 89
nom recherché Mitenne     Laurence

. do cherche
telephone (8chiffres):55 66 77 88
ce numéro ne figure pas au fichier
```

écran 15.6 : programme (amélioré) de recherche d'un abonné de numéro donné

4. QUELQUES RÈGLES À PROPOS DE LA STRUCTURE DE CHOIX

Il faut bien veiller à placer les mots ELSE et ENDIF, seuls sur une ligne. Il en va de même pour le mot IF accompagné de sa condition.

Vous pouvez placer autant de commandes que vous le souhaitez dans chaque partie du choix (entre IF et ELSE d'une part, entre ELSE et ENDIF d'autre part).

Vous pouvez, si vous le souhaitez, omettre la partie relative au cas où la condition est fausse, comme dans cet exemple :

```
IF EOF
   ? ''fin de fichier atteinte''
ENDIF
```

▷ **Entraînez-vous**

(Voir correction en fin de volume).

- (1) Réalisez un programme recherchant dans le fichier STOCK un produit dont on lui fournit la référence. Si le produit existe, le programme en affichera la cétégorie, la quantité et la valeur. Dans le cas contraire, il affichera un message ad'hoc.

5. LA STRUCTURE DE CHOIX MULTIPLE : DO CASE... ENDCASE

Nous avons vu comment la structure de choix permet de ''choisir'' entre deux possibilités. Mais il existe des circonstances qui nécessitent de pouvoir choisir entre plus de deux possibilités. Dbase possède une telle structure que nous allons vous présenter sur un exemple simple, comme nous l'avions fait pour la structure de choix.

Voyez ce programme, que nous avons nommé CHOIXMUL :

```
set talk off
input "donnez un nombre" to n
do case
  case n<10
    ? "votre nombre est inférieur à 10"
  case n>=10 .and. n <=20
    ? "votre nombre est compris entre 10 et 20"
  case n>20
    ? "votre nombre est supérieur à 20"
endcase
? "au revoir et merci"
set talk on
```

écran 15.7 : la structure de choix multiple

Nous y utilisons une nouvelle structure introduite par **DO CASE** et terminée par **ENDCASE**. Entre ces deux lignes, chaque partie du choix

est introduite par le mot CASE suivi de la condition du choix. Ici, nous avons prévu trois possibilités :

1) le nombre fourni (N) est inférieur à 10

2) le nombre fourni est compris entre 10 et 20 (ces valeurs incluses)

3) le nombre fourni est supérieur à 20

Exécutons ce programme à diverses reprises :

```
. do choixmul
donnez un nombre:78
votre nombre est supérieur à 20
au revoir et merci

. do choixmul
donnez un nombre:5
votre nombre est inférieur à 10
au revoir et merci

. do choixmul
donnez un nombre:14
votre nombre est compris entre 10 et 20
au revoir et merci
```

écran 15.8 : exemples d'exécution du programme de l'écran 15.7

Remarques :

Ici, nous avons prévu trois possibilités mais leur nombre peut être quelconque. D'autre part, dans notre exemple, parmi les trois conditions mentionnées, il y en a toujours une et seule qui est satisfaite. C'est la une situation courante mais qui n'est pas du tout imposée par Dbase. Rien n'empêche que, dans certains cas, aucune des conditions ne soit satisfaite : Dbase passe alors simplement à la commande suivant le ENDCASE. Si, par contre, plusieurs conditions se trouvent vérifiées, Dbase exécute les commandes correspondantes au premier cas trouvé satisfaisant. Les exemples de ''l'Entrainez-vous'' ci-dessous illustrent ces diverses situations.

▷ **Entraînez-vous**

- Exécutez ce programme avec les réponses : 5,25 puis 10 :

  SET TALK OFF

  INPUT ''nombre'' TO N

```
DO CASE
   CASE N < 10
      ? ''petit nombre''
   CASE N > 10
      ? ''grand nombre''
ENDCASE
SET TALK ON
```

- Exécutez ce programme avec les réponses 6, 15, 25 puis 10 :

```
SET TALK OFF
INPUT ''nombre'' TO N
DO CASE
   CASE N < 20
      ? ''petit nombre''
   CASE N > 10
      ? ''grand nombre''
   CASE N = 5
      ? ''vous avez dit 5''
      ? ''mais je n'ai rien vu !''
ENDCASE
SET TALK ON
```

6. POUR LES ''IMPRÉVUS'' : OTHERWISE

Nous avons vu que, dans la structure de choix multiple, lorsqu'aucune des conditions n'est satisfaite, Dbase passe simplement ''à la suite'', c'est-à-dire à la commande qui suit la ligne ENDCASE. En fait, il vous est possible de demander à Dbase d'exécuter dans ce cas certaines commandes de votre choix. Il suffit pour cela de les placer à la fin de la structure de choix multiple, en les faisant précéder de **OTHERWISE**.

Voyez cet exemple :

```
set talk off
input "donnez un nombre" to n
do case
  case n<10
    ? "petit nombre"
  case n>10
    ? "grand nombre"
  otherwise
    ? "vous avez dit 10"
endcase
set talk on
```

```
. do other
donnez un nombre:5
petit nombre
. do other
donnez un nombre:25
grand nombre
. do other
donnez un nombre:10
vous avez dit 10
```

écran 15.9 : pour les imprévus : OTHERWISE

7. UN PROGRAMME "A MENU"

Nous vous proposons un programme (voir écran 15.10 page) permettant à son utilisateur d'effectuer l'une des tâches suivantes sur le fichier STOCK :

— liste suivant les catégories, classées par ordre alphabétique

— liste suivant les références

— liste suivant les prix

Pour simplifier quelque peu le programme, nous avons supposé que les trois fichiers index correspondants existaient et étaient à jour. Dans un cas réel, il pourrait être nécessaire de les créer suivant les besoins.

```
set talk off
erase
use stock
? "que souhaitez vous"
? " 1 - liste par catégorie"
? " 2 - liste par référence"
? " 3 - liste par prix"
accept "votre choix" to choix
do case
  case choix="1"
    set index to cstock
    list
  case choix="2"
    set index to rstock
    list
  case choix="3"
    set index to pstock
    list
  otherwise
    ? "vous avez fait un mauvais choix"
endcase
set talk off
```

écran 15.10 : un programme à menu permettant différentes listes du fichier STOCK

Notez que la deuxième ligne comporte une commande nouvelle :

ERASE

Celle-ci ne fait rien d'autre qu'effacer l'écran. Celà nous permet d'afficher clairement le choix proposé à l'utilisateur, sous forme de ''menu''.

Voici trois exemples d'exécution de ce programme :

```
. do menu
que souhaitez vous
 1 - liste par catégorie
 2 - liste par référence
 3 - liste par prix
votre choix:1
00001  Cafetière 12 T        A 432    137.47     32
00003  Grille pain           A 521    158.75     23
00004  Four à raclette 6 P   A 427    133.72     15
00005  Friteuse 1 kg         B 433    349.25     21
00002  Four à micro-ondes    E 248   1875.25      7
00006  Table de cuisson      E 647   2450.00      6

. do menu
que souhaitez vous
 1 - liste par catégorie
 2 - liste par référence
 3 - liste par prix
votre choix:3
00004  Four à raclette 6 P   A 427    133.72     15
00001  Cafetière 12 T        A 432    137.47     32
00003  Grille pain           A 521    158.75     23
00005  Friteuse 1 kg         B 433    349.25     21
00002  Four à micro-ondes    E 248   1875.25      7
00006  Table de cuisson      E 647   2450.00      6

. do menu
que souhaitez vous
 1 - liste par catégorie
 2 - liste par référence
 3 - liste par prix
votre choix:5
vous avez fait un mauvais choix
```

écran 15.11 : exemples d'exécution du programme de l'écran 15.10

Note : en fait, l'écran s'efface après chaque commande DO.

Remarques :

1) Nous avons choisi de placer le choix de l'utilisateur dans une variable de type caractère, bien que la valeur attendue soit un chiffre (commande ACCEPT et non INPUT). Cela nous permet d'éviter à l'utilisateur de se voir gratifier d'un message d'erreur (en provenance de Dbase) dans le cas où il répondrait autrement que sous forme numérique. Dans des programmes réels destinés à des néophytes, il est vivement conseillé de procéder ainsi.

2) Lorsque l'utilisateur fait un ''mauvaix choix'', le programme s'interrompt simplement après le lui avoir signalé, sans lui fournir de ''nouvelle chance''. Nous verrons plus tard comment remédier à ce problème.

3) La même commande LIST est écrite en trois endroits différents. Il pourrait paraître plus judicieux de la placer après la structure de choix multiple. Malheureusement, dans ce cas, elle se trouverait également exécutée en cas de ''mauvais choix'' !

8. POUR Y VOIR PLUS CLAIR DANS VOS PROGRAMMES : LES COMMENTAIRES

Lorsque vous serez amenés à réaliser de nombreux programmes, plus ou moins complexes, vous serez très vite confrontés à des problèmes tels que ceux-ci :

— quel est le rôle d'un programme donné.

— quelles sont ses conditions d'utilisation (par exemple, les fichiers nécessaires à son exécution)

— que font chacunes des commandes qui le constituent. Ce dernier point sera particulièrement crucial lorsque vous chercherez à modifier (ou même simplement à ''mettre au point'') un programme existant.

Comme tous les langages de programmation, Dbase vous autorise à placer des ''commentaires'' dans vos programmes. Ce sont des lignes particulières contenant des explications destinées uniquement à celui qui examine le programme (et non à celui qui l'utilise). Ces lignes qui seront complètement ignorées de Dbase doivent commencer par le caractère *.

A titre d'exemple, voici une façon (parmi tant d'autres) d'introduire des commentaires dans notre programme précédent. (Nous ne vous donnons pas d'autres exemples d'exécution puisque ces commentaires, ignorés de Dbase, ne modifient en rien son comportement).

```
* ***** PROGRAMME DE LISTE DU FICHIER STOCK *****
*       - par catégorie
*       - par référence
*       - par prix
*
* Attention : les 3 fichiers index CSTOCK, RSTOCK
*                 et PSTOCK doivent exister

* ---------- initialisations --------------------
set talk off
erase
use stock
*
* ---------- présentation du menu ---------------
*         et lecture du choix de l'utilisateur
? "que souhaitez vous"
? " 1 - liste par catégorie"
? " 2 - liste par référence"
? " 3 - liste par prix"
accept "votre choix" to choix
*
* ---------- selection du traitement en ---------
*               fonction du choix
do case
  case choix="1"
    set index to cstock
    list
  case choix="2"
    set index to rstock
    list
  case choix="3"
    set index to pstock
    list
  otherwise
    ? "vous avez fait un mauvais choix"
endcase
*
* ---------- fin du programme -------------------
set talk off
```

écran 15.12 : un programme ''commenté''

XVI Pour répéter des commandes : la structure de boucle

Nous avons appris comment, au sein d'un programme, demander à Dbase de faire un ''choix'' entre différents ensembles de commandes. Nous disposions pour cela de deux ''structures de choix'' mises en place par IF... ENDIF ou DO CASE... ENDCASE.

Néanmoins, chaque commande d'un programme se trouvait exécutée au maximum *une fois*. Or, il est fréquent de devoir faire exécuter *plusieurs fois* de suite un même ensemble de commandes : par exemple, pour effectuer le même traitement sur tous les enregistrements d'un fichier. Il paraît alors plus que souhaitable de ne pas devoir écrire plusieurs fois les instructions en question (méthode difficilement applicable pour traiter 1 000 enregistrements, et inexploitable pour en traiter un nombre non connu à l'avance !).

Dbase nous offre une solution à ce problème avec la structure de boucle (ou répétition) mise en place par les commandes **DO WHILE** et **ENDDO**.

1. LA STRUCTURE DE BOUCLE : DO WHILE... ENDDO

Comme nous l'avons fait pour les structures de choix, nous allons

commencer par l'introduire sur un exemple simple. Créons ce petit programme nommé : BOUCLE

```
set talk off
*
store "O" to rep
do while rep="O"
  ? "bonjour cher ami"
  accept "on continue " to rep
  ? "merci de votre réponse"
enddo
? "au revoir"
*
set talk on
```

écran 16.1 : la structure de boucle.

Les lignes suivantes :

DO WHILE REP = ''O''

_____

_____

ENDDO

signifient : répéter toutes les commandes situées entre DO WHILE et ENDDO, *tant que* (while) la condition mentionnée (ici REP = ''O'') est vraie.

Qu'entendons-nous par ''tant que'' ? En fait, lorsque Dbase rencontre la commande DO WHILE, il examine la condition qui lui est associée. Si elle est vraie, il exécute les commandes suivantes (jusqu'à ENDDO) puis il examine à nouveau la condition, et ainsi de suite.

Lorsque, lors de l'un de ces examens, Dbase trouve que la condition est fausse, il passe à l'exécution de la commande située après ENDDO.

Confirmons tout cela en exécutant une fois notre programme (nommé BOUCLE) :

```
. do boucle
bonjour cher ami
on continue :O
merci de votre réponse
bonjour cher ami
on continue :O
merci de votre réponse
bonjour cher ami
on continue :bof
merci de votre réponse
au revoir
```

écran 16.2 : exécution du programme de l'écran 16.1

Remarques :

1) Il est très important de noter que Dbase n'examine la condition associée à DO WHILE (ici REP = ''O'') qu'avant chaque (éventuel) nouveau tour de boucle, et uniquement à ce moment-là. Autrement dit, lorsque au sein de la boucle, la condition passe de vrai à faux, Dbase ne l'interrompt pas immédiatement. Il continue à exécuter les commandes suivantes (s'il y en a) jusqu'au ENDDO puis, et seulement à ce moment-là, examine la condition et découvre qu'elle est fausse... Ainsi, dans notre exemple le message ''merci de votre réponse'' est affiché lors du dernier tour de boucle.

2) Ici, la condition était REP = ''O'' ; il est clair que cette condition est fausse pour toute valeur de REP différente de ''**O**''. De sorte que ''**N**'' ou ''**n**'' ou ''**o**'' ou ''**bof**'' sont des réponses faisant devenir fausse la condition. Par contre, les réponses ''**OUI**'', ''**Oui**'' provoqueront un résultat dépendant de la valeur de l'indicateur EXACT.

3) La commande : STORE ''O'' TO REP est indispensable pour que Dbase puisse évaluer la condition REP = ''O'' lorsqu'il aborde la boucle pour la première fois.

4) Si la condition est fausse lorsque Dbase aborde la ligne DO WHILE pour la première fois, les commandes situées dans la boucle ne sont pas exécutées.

▷ **Entraînez-vous**

(Les points comportant un numéro sont corrigés en fin de volume).

- Voyez comment se comporte Dbase lorsque vous supprimez, dans notre précédent programme BOUCLE, la ligne :

  STORE ''O'' TO REP

- Cherchez à prévoir ce que fera ce programme :

```
SET TALK OFF
STORE 1 TO N
DO WHILE N <> 0
   ? "hello"
ENDDO
SET TALK ON
```

Vérifiez vos hypothèses. Interrompez l'exécution de votre programme (par la touche EXIT sur PCW ou ESC sur CPC). Voyez à l'aide de la commande DISPLAY STATUS comme l'indicateur TALK est resté désactivé (OFF) (et ceci, parce que la commande SET TALK ON de votre programme n'a pas été exécutée). Redonnez-lui le bon état.

(1) Réalisez un programme qui demande un nombre à l'utilisateur, en l'interrogeant tant que ce nombre n'est pas 25. A titre d'exemple, son exécution se présenterait ainsi :

```
. do devin
choisissez un nombre :5
choisissez un nombre :86
choisissez un nombre :32
choisissez un nombre :13
choisissez un nombre :25
bravo - c'est ça
```

2. POUR RÉPÉTER UN TRAITEMENT SUR TOUS LES ENREGISTREMENTS D'UN FICHIER

Supposez que vous ayez écrit des commandes permettant de réaliser un certain traitement sur un enregistrement d'un fichier et que vous souhaitiez maintenant appliquer ce traitement à tous les enregistrements du fichier. Bien entendu, vous songez à inclure ces commandes dans une structure de boucle.

La condition associée doit alors être : fin de fichier non rencontrée, c'est-à-dire :

.NOT.EOF

Néanmoins, cela ne suffit pas car, dans ces conditions, le traitement ainsi répété porterait toujours sur le même enregistrement. Il faut donc, en outre, prévoir dans la boucle de faire progresser le pointeur par SKIP. D'autre part, il est nécessaire que ce pointeur soit convenablement

initialisé avant l'entrée dans la boucle, ce qui sera le cas si le fichier vient d'être "ouvert" par USE.

D'où un "schéma type" correspondant au traitement de tous les enregistrements d'un fichier :

```
USE fichier
DO WHILE .NOT. EOF
        ____________                 traitement de
        ____________   }             l'enregistrement
        ____________   }                courant
        SKIP
ENDDO
```

tableau 16.3 : pour traiter tous les enregistrements d'un fichier

A titre d'exemple, voici un programme affichant des informations à partir du fichier REPERT.

```
********** PROGRAMME D'AFFICHAGE DU REPERTOIRE **********
*
* ---------- initialisations ----------------------------
set talk off
use repert
*
* ---------- boucle de traitement de tous les enreg -----
do while .not. eof
  ? "Je m'appelle      : ", nom
  ? "  J'habite a ", ville, " et je mesure", taille, "m."
  ?
  skip
enddo
*
* ---------- fin programme ------------------------------
set talk on
```

écran 16.4 : programme d'affichage de REPERT

```
. do affrep
Je m'appelle      :  Thomas
  J'habite a  NEVERS    et je mesure  1.75 m.

Je m'appelle      :  Mitenne
  J'habite a  FREJUS    et je mesure  1.58 m.

Je m'appelle      :  Loiseau
  J'habite a  MORREZ    et je mesure  1.71 m.

Je m'appelle      :  Meunier
  J'habite a  PARIS     et je mesure  1.83 m.

Je m'appelle      :  Taillefert
  J'habite a  SULLY     et je mesure  1.78 m.

Je m'appelle      :  Grillon
  J'habite a  PARIS     et je mesure  1.68 m.
```

écran 16.5 : exécution du programme de l'écran 16.4

Remarque

La commande DISPLAY permettrait d'obtenir les mêmes informations, avec cependant une présentation différente (nous ne pourrions pas disposer les informations d'un enregistrement sur deux lignes et les séparer par une ligne blanche).

3. POUR VOUS AIDER À DÉTECTER VOS ERREURS

Dès que vos programmes deviennent quelque peu complexes, il peut vous être difficile de diagnostiquer la cause d'un fonctionnement anormal. Dbase vous offre quelques possibilités pour vous aider dans votre recherche.

SET TALK ON

Généralement, on a tendance, dans un programme, à supprimer les "compte-rendus" de Dbase. Cependant, en cas de difficulté, vous obtiendrez des renseignements utiles en "réactivant" cette option dans les parties "douteuses" de votre programme.

SET ECHO ON

Cette commande demande à Dbase de vous afficher systématique-

ment toutes les commandes qu'il exécute. Bien entendu, cette possibilité est utilisable en ''mode direct'', mais elle n'y présente guère d'intérêt dans la mesure où elle amène Dbase à reproduire textuellement chacune des commandes que vous lui fournissez avant de l'exécuter.

En ''mode programme'', par contre, vous pouvez ainsi suivre à la trace le déroulement de votre programme.

Voici un exemple de cette possibilité appliquée à notre programme ''BOUCLE'' (présenté au début de ce chapitre)

```
. set echo on
. do boucle
do boucle
set talk off
store "O" to rep
do while rep="O"
  ? "bonjour cher ami"
bonjour cher ami
  accept "on continue " to rep
on continue :O
  ? "merci de votre réponse"
merci de votre réponse
enddo
do while rep="O"
  ? "bonjour cher ami"
bonjour cher ami
  accept "on continue " to rep
on continue :bof
  ? "merci de votre réponse"
merci de votre réponse
enddo
do while rep="O"
? "au revoir"
au revoir
set talk on
. set echo off
set echo off
.
```

écran 16.6 : pour obtenir une trace de l'exécution d'un programme : SET ECHON ON

Notez que, à l'écran, s'entrelacent à la fois les messages en provenance de votre programme et la trace des commandes du programme.

SET STEP ON

Cette commande demande à Dbase d'exécuter le programme ''pas à pas'', c'est-à-dire en fait ligne par ligne. A chaque pas, vous pouvez choisir entre trois possibilités :

— poursuivre l'exécution
— passer une commande au clavier. Vous pouvez ainsi, par exemple, modifier la valeur d'une variable,...
— annuler l'exécution (n'oubliez pas, sur PCW, ESC correspond à la touche EXIT).

Voici un exemple de cette possibilité appliquée à notre programe ''BOUCLE''. Voyez comme, bien qu'ayant répondu ''N'' à la question ''on continue'', nous avons ultérieurement passé une commande

STORE ''O'' TO REP

pour modifier la valeur de la variable REP. Vous constatez qu'alors la boucle a bien été parcourue une seconde fois.

```
. set step on
. do boucle
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
bonjour cher ami
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
on continue :N
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
merci de votre réponse
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? N
. store "O" to rep
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
bonjour cher ami
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
on continue :bof
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
merci de votre réponse
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
au revoir
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
ETAPE SUIVANTE --> Y=OUI : N=CMDE CLAVIER : ESC=ANNULATION ? Y
```

écran 16.7 : pour exécuter un programme pas à pas : SET STEP ON

Remarques

Nous vous avons présenté les trois options, de manière indépendante les unes des autres. Mais il est, bien entendu, possible de les conjuguer. D'autre part, lorsque vous suspectez une partie précise d'un programme, il vous est toujours possible de n'activer une option (TALK, ECHO, STEP) que pour cette partie du programme ; il vous suffit de placer la commande d'activation au sein de votre programme (au lieu de l'exécuter en mode direct avant d'exécuter le programme).

XVII Pour bien gérer l'écran

Dans les programmes que nous avons réalisés jusqu'ici, nous laissions Dbase ''gérer'' l'affichage à l'écran. Que nous utilisions les commandes d'affichage (?, DISPLAY,...) ou de saisie (INPUT, ACCEPT), Dbase travaillait toujours *ligne après ligne*. Chaque nouvel affichage se faisait à la suite du précédent.

Nous allons maintenant apprendre à gérer nous même l'écran, en y disposant les informations aux emplacements qui nous conviennent. Pour ce faire, nous allons introduire de nouvelles commandes : à... SAY..., à... SAY... GET... et READ.

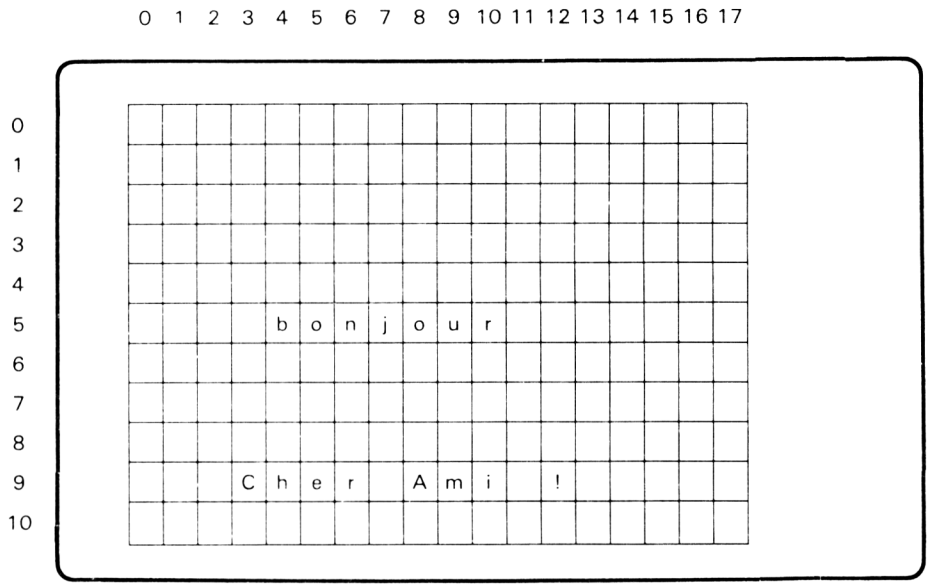
1. POUR AFFICHER À VOTRE GUISE : à... SAY...

Commencez par créer et exécuter ce petit programme.

```
erase
à 5,4 say "bonjour"
à 9,3 say "Cher  Ami !"
```

écran 17.1 : pour gérer l'affichage : à SAY

Vous constatez que son exécution efface d'abord l'écran et y affiche les textes ''bonjour'' et ''Cher Ami'' disposés comme dans ce schéma.



écran 17.2 : la grille d'écran

Nous y avons représenté la partie supérieure gauche de l'écran. Chaque case correspond à un emplacement où peut s'afficher un caractère. Le nombre d'emplacements dépend du type d'Amstrad que vous possédez. Chacun d'entre eux est repéré par :

— un numéro de ligne allant de 0 à 31
— un numéro de colonne allant de 0 à 82 sur PCW et de 0 à 79 sur CPC.

Une commande telle que :

à 5,4 SAY ''bonjour''

signifie : à partir de l'emplacement 5,4 (ligne 5, colonne 4), afficher l'expression : ''bonjour''.

2. UN PROGRAMME DE LISTE ''PLEIN ÉCRAN''

Voici, à titre d'illustration, un programme affichant chaque enregistrement du fichier REPERT sur l'ensemble de l'écran.

Pour éviter que l'utilisateur ne voie défiler tous les enregistrements sans pouvoir les lire, nous avons prévu une attente après affichage. Plus précisément, l'utilisateur doit simplement taper ''RETURN'' pour voir apparaître l'enregistrement suivant.

```
* ************ LISTE DE REPERT EN PLEIN ECRAN ************
*
* ---------- initialisations ----------------------------
set talk off
use repert
* ---------- boucle de traitement de tous les enreg ------
do while .not. eof
  erase
  à 2,15  say "ENREGISTREMENT NUMERO "
  à 2,40  say #
  à 10,5  say "NOM"
  à 10,16 say nom
  à 12,5  say "PRENOM"
  à 12,16 say prenom
  à 10,35 say "CODE POSTAL"
  à 10,50 say codep
  à 12,35 say "VILLE"
  à 12,50 say ville
  à 15,5  say "TAILLE EN M."
  à 15,25 say taille
  à 20,20 say "NUMERO DE TELEPHONE"
  à 20,41 say tele
  à 29,40 say "pour voir la suite, tapez sur RETURN"
  accept to nul
  skip
enddo
* ---------- fin programme -------------------------------
set talk on
```

écran 17.3 : liste du fichier REPERT en ''plein écran''

▷ **Entraînez-vous**

- Utilisez ce petit programme pour expérimenter ce que fait la commande à ... SAY avec des numéros de ligne ou de colonne supérieurs aux limites prévues.

```
set talk off
input "numero ligne" to nl
input "numero colonne" to nc
erase
à nl,nc say "hello"
```

3. POUR PROGRAMMER VOS SAISIES EN MODE "PLEIN ÉCRAN" : GET ET READ

Nous venons de voir cmment, en quelque sorte, afficher des informations en mode "plein écran". Mais Dbase vous permet également de définir vos propres "masques de saisie", grâce aux instructions :

à... SAY... GET...

et

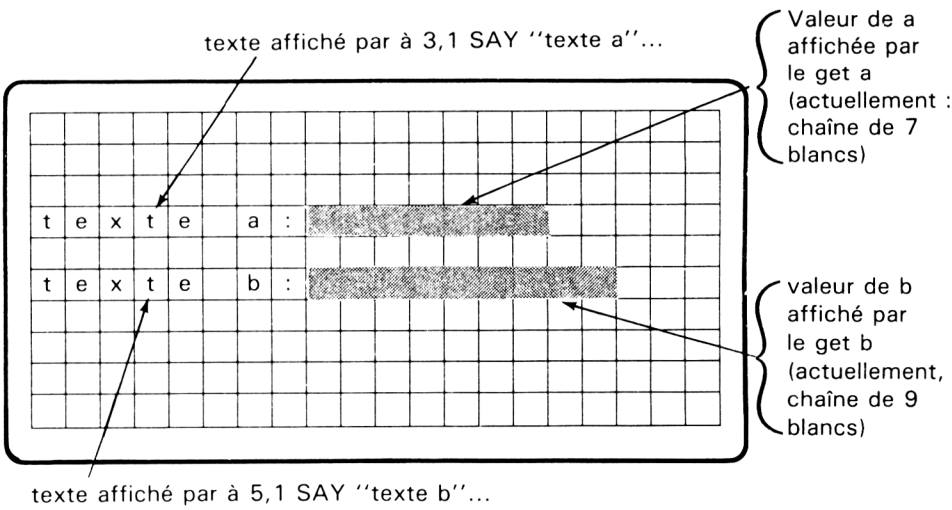
READ

Essayez ce petit programme qui a pour but de saisir deux informations de type chaîne dans les variables a et b :

```
set talk off
erase
store "        " to a
store "          " to b
à 3,1 say "texte a" get a
à 5,1 say "texte b" get b
read
à 10,1 say "vous avez fourni "
? a, "et      ", b
set talk on
```

écran 17.4 : les commandes de saisie : à... SAY... GET et READ

Lorsque vous cherchez à l'exécuter, vous constatez que l'écran se présente ainsi :



écran 17.5 : le "masque de saisie" proposé par le programme de l'écran 17.4

Les quatre premières lignes du programme sont classiques (nous reviendrons plus loin sur la nécessité des commandes STORE). La ligne :

à 3,1 SAY ''texte a'' GET A

demande deux choses à Dbase :

1) afficher à l'endroit indiqué la chaîne mentionnée à la suite de SAY (ici ''texte a'').

2) prévoir qu'il faudra, à la suite, réaliser ''une saisie'' d'une valeur pour une variable nommée A. Pour l'instant, Dbase affiche l'actuel contenu de cette variable (une chaîne formée de 7 blancs) en *inversion vidéo*. C'est dans cette zone en inversion vidéo que l'utilisateur pourra venir fournir une nouvelle valeur.

Mais, pour l'instant, Dbase n'effectue aucune saisie

La ligne suivante :

à 5,1 SAY ''texte b'' GET B

réalise des choses comparables avec la variable B.

C'est **la commande READ** qui **déclenche effectivement le processus de saisie.** Pour ce faire, Dbase place le pointeur sur le début de la première zone et vous laisse faire les modifications comme avec une commande EDIT. Notamment, vous pouvez changer de zone (en avant ou en arrière). Dbase considère la saisie achevée, soit lorsque la dernière zone est remplie soit lorsque vous tapez ''RETURN'' ou → alors que le curseur s'y trouve.

Remarque

Il est nécessaire que la variable mentionnée à la suite de GET *existe.* (Elle ne peut pas être créée par GET). C'est d'ailleurs son contenu qui sert à définir la taille de la zone destinée à la saisie.

▷ **Entraînez-vous**

- Essayez, à diverses reprises, le petit programme précédent en vous familiarisant bien avec le mécanisme de saisie.

- Ajoutez deux commandes de part et d'autre de la commande READ, comme ceci :

  à 15,1 SAY "on va saisir les informations"
  READ
  à 15,1 SAY "on a saisi les informations"

L'exécution de ce nouveau programme doit vous confirmer que c'est bien la commande READ qui déclenche la saisie.

- Inversez les emplacements de saisie des variables A et B, en modifiant ainsi les deux commandes correspondantes

  à 5,1 SAY "texte a" GET A
  à 3,1 SAY "texte b" GET B

Voyez comme c'est l'ordre des commandes qui impose l'ordre des saisies.

- Voyez comment se déroule la saisie d'une variable numérique en ajoutant ces commandes :

  STORE 25 to N
  à 7,1 SAY "valeur" GET N

  (Notez comme la zone de saisie est de longueur 10, quelle que soit la valeur initiale de N)

- Voyez ce qui se produit lorsqu'une variable nommée après GET n'existe pas. (Pour cela, supprimez une commande STORE et pour être certain que la variable en question n'existe pas, faites RELEASE ALL avant d'exécuter le programme).

- Revenez au programme initial et voyez ce qui se produit si vous ajoutez une commande d'effacement d'écran (ERASE) immédiatement avant READ.

- Revenez au programme initial et voyez ce qui se passe lorsque vous oubliez la commande READ.

- Revenez au programme initial et insérez une (seconde) commande READ entre les deux commandes à... SAY... GET... Voyez comme il y a, cette fois, deux phases de saisie.

4. POUR CONTRÔLER VOS SAISIES

4.1. La méthode

Lorsque nous saisissons des informations (par APPEND ou EDIT), Dbase n'effectue aucun contrôle, à l'exception du type, en cas de champ numérique. Or, il existe de nombreuses circonstances où l'information à entrer dans un champ ne peut prendre toutes les valeurs possibles.

Ainsi, par exemple, dans notre fichier stock, nous pourrions souhaiter éviter que l'utilisateur ne puisse entrer de valeur autre que A, B, C ou E. Cela est assez facile à réaliser par programme, à partir du moment où vous gérez vous même la saisie.

Cependant, si nous avons appris à saisir par programme une information depuis le clavier, nous n'avons pas vu comment la placer dans un enregistrement donné.

En fait, si un tel enregistrement existe, il suffit d'y placer le pointeur (par GOTO, FIND,...) et d'utiliser une ou plusieurs commandes REPLACE (Notez qu'il serait stupide d'incorporer une commande EDIT dans un programme, puisqu'alors vous ne contrôleriez plus la saisie).

Si un tel enregistrement n'existe pas, il n'est bien sûr pas question d'utiliser APPEND (là encore, vous ne contrôleriez plus la saisie). Il faut donc créer un nouvel enregistrement contenant l'information voulue. Pour ce faire, Dbase vous offre une nouvelle commande :

APPEND BLANK

Celle-ci ajoute un enregistrement vide au fichier courant. Comment le remplir ? Simplement à l'aide de REPLACE.

Nous vous proposons, à titre d'exemple, de réaliser un programme d'extension du fichier STOCK, dans lequel (par souci de simplification) nous vous limitons à la saisie des informations référence et catégorie (avec contrôle de celle-ci). Avant de voir le programme complet correspondant, voyons d'abord comment pourraient se présenter les commandes de saisie.

4.2. Les commandes de saisie avec contrôle

```
* ---------- préparation de la saisie ----------------
store " " to ca
store "   " to rf
store "N" to ok
* ---------- saisie jusqu'à catégorie OK -------------
do while ok<>"O"
  erase
  à 2,1 say "catégorie (A/B/C/E)" get ca
  à 5,1 say "référence           " get rf
  read
  if ca="A" .or. ca="B" .or. ca="C" .or. ca="E"
      store "O" to ok
    else
      à 15,5 say "***** CATEGORIE INCORRECTE *****"
      accept "       pour recommencer, tapez RETURN" to nul
  endif
enddo
```

écran 17.6 : saisie des informations catégorie et référence avec contrôle de la catégorie

Notez tout d'abord les noms de variables que nous avons choisi pour effectuer la saisie : CA (pour la catégorie) et RF (pour la référence).

Voyez comme nous répétons les commandes de saisie tant que la catégorie n'est pas correcte. Pour ce faire, nous utilisons une variable nommée OK. Celle-ci est initialisée à ''N'' et nous lui attribuons la valeur ''O'' quand la catégorie est satisfaisante.

▷ **Entraînez-vous**

- Vérifiez que les commandes précédentes fonctionnent convenablement en exécutant le programme proposé (vous pouvez provisoirement ajouter des commandes, par exemple : SET TALK OFF, SET TALK ON).

4.3. Un programme d'extension du fichier stock

Il ne nous reste plus qu'à répéter cet ensemble de commandes pour chaque enregistrement que l'utilisateur souhaite créer, en le complétant par des commandes APPEND BLANK et REPLACE.

Auparavant, nous voyons qu'il est nécessaire d'inclure toutes ces commandes dans une boucle DO WHILE, laquelle nécessite une condition. Encore nous faut-il décider de la manière dont l'utilisateur signalera qu'il n'a plus d'enregistrements à ajouter. Nous pouvons convenir qu'il le fasse en fournissant un ''blanc'' pour la catégorie. Notre boucle se présenterait donc ainsi :

```
DO WHILE CA<> '' ''
    ________        }   instructions de saisie et
    ________        }   de création d'un nouvel
    ________        }       enregistrement
ENDDO
```

Toutefois, si nous procédons comme cela, nous constaterons qu'un dernier enregistrement vide se trouvera ajouté au fichier, ce qui n'est guère satisfaisant. Pour l'entrer, nous procèderons comme ceci :

```
DO WHILE CA <> '' ''
    ________        }   instructions de saisie
    ________        }

IF CA <> '' ''
        ______      }   instructions de création
        ______      }       d'un nouvel
                        enregistrement
ENDIF
ENDDO
```

Enfin, il nous faut modifier les commandes de saisie pour que le blanc n'entraîne pas le ''rejet'' de cette catégorie.

D'où le programme définitif :

```
****** PROGRAMME D'EXTENSION DU FICHIER STOCK ********
*
* ---------- initialisations -------------------------
set talk off
use stock
store "x" to ca
*
* ---------- boucle de traitement des enregistrements
*                     à ajouter      -----------------
do while ca<> " "
  * ---------- préparation de la saisie ----------------
  store " " to ca
  store "   " to rf
  store "N" to ok
  * ---------- saisie jusqu'à catégorie OK -------------
  do while ok<>"O"
    erase
    à 2,1 say "catégorie (A/B/C/E)" get ca
    à 5,1 say "référence          " get rf
    read
    if ca="A" .or. ca="B" .or. ca="C" .or. ca="E" .or. ca=" "
        store "O" to ok
      else
        à 15,5 say "***** CATEGORIE INCORRECTE *****"
        accept "      pour recommencer, tapez RETURN" to nul
    endif
  enddo
  * --------- création d'un nouvel enregistrement
  *               s'il y a lieu    ---------------------
  if ca<>" "
    append blank
    replace cat with ca
    replace ref with rf
  endif
enddo
*
* ---------- fin programme ------------------------------
use
set talk on
```

écran 17.7 : un programme d'extension du fichier STOCK avec contrôle de la catégorie

XVIII A vous de jouer

L'objectif essentiel de ce livre était de vous initier à l'utilisation de DBase II sur AMSTRAD. Pour y parvenir, nous avons cherché à vous exposer progressivement les notions fondamentales intervenant dans un tel logiciel. Pour ne pas alourdir inutilement notre démarche, nous avons délibéremment choisir d'ignorer certains aspects où certaines commandes qui, à notre avis, ne s'imposaient pas dans une phase d'apprentissage.

Si vous avez réalisé l'ensemble des ''Entraînez-vous'' (du moins, ceux des 13 Premiers chapitres), vous devez être maintenant parvenu à un stade où vous pouvez ''voler de vos propres ailes''. En particulier, vous devez être en mesure d'imaginer les manipulations appropriées vous permettant d'expérimenter de nouvelles possibilités.

Enfin et surtout, vous devez maintenant avoir une attitude adulte devant le logiciel et aborder la gestion de vos propres bases de données.

D'autre part, nous vous conseillons de poursuivre (tranquillement) l'étude des possibilités qui n'ont pas été abordées ici. Pour vous y aider, nous vous fournissons ici quelques indications sur les points que nous n'avions pas considéré jusqu'ici.

1 - La commande BROWSE

Elle permet de considérer n'importe quel fichier comme un tableau rectangulaire dans lequel les lignes correspondent aux enregistrements et les colonnes aux champs. Votre écran devient alors une sorte de ''fenêtre'' que vous pouvez déplacer (horizontalement et verticalement) sur ce tableau. Elle autorise les ''mise à jour'' du fichier (modification, ajout, marquage pour effacement).

2 - La commande CHANGE... FIELDS

Elle permet d'effectuer des modifications de certains champs d'un fichier ; pour ce faire, elle ne présente que les valeurs des champs concernés en demandant de fournir les valeurs de remplacement.

3 - Les Macros

Elles ont essentiellement deux intérêts :

a) elles permettent d'utiliser le contenu d'une variable dans une commande. Par exemple, si la variable L contient la chaîne

''INDEX NREPERT''

la commande :

USE REPERT &L

sera équivalente à la commande :

USE REPERT INDEX NREPERT

b) Elles permettent d'employer la commande FIND, au sein d'un programme, pour retrouver une clé dont la valeur figure dans une variable. Ainsi, si NOM est une variable, la commande

FIND &NOM

permettra de retrouver l'enregistrement ayant comme clé la valeur contenue dans la variable NOM (Cela serait impossible, sans la notion de macro).

4 - Conservation des valeurs des variables

SAVE TO sauvegarde dans un fichier (de type MEM) toutes les variables existant au moment où cette commande est exécutée (nom, type, valeur).

RESTORE FROM permet de retrouver des variables ainsi sauvegardées.

5 - USING et PICTURE

Le paramètre USING, associé à la commande à..., SAY..., permet de ''contrôler le format d'affichage'' de l'information correspondante.

Le paramètre PICTURE, associé à la commande à... GET, permet de ''contrôler la saisie'' de l'information correspondante.

6 - Les fonctions

Outre celles déjà rencontrées, il existe :

INT	partie entière d'une expression numérique
STR	Conversion d'une expression numérique en une chaîne de caractères
VAL	conversion d'une chaîne de caractères en numérique
LEN	longueur d'une chaîne de caractères
*	indique si l'enregistrement courant a été marqué pour effacement
@	recherche l'occurence d'une chaîne dans une autre
DATE ( )	fournit la date indiquée lors du lancement de Dbase
FILE	teste l'existence d'un fichier
TYPE	fournit le type d'une expression

7 - Les indicateurs qu'il est possible d'activer par SET

La commande DISPLAY STATUS vous en fournit la liste complète.

8 - Zone primaire et zone secondaire

DBase II vous permet d'utiliser deux zones de travail nommées PRIMARY (pour celle qui est employée par défaut) et SECONDARY. Chacune d'entre elles peut être associée à un fichier (par USE) avec d'éventuels index et dispose de son propre pointeur. Pour ''passer'' d'une zone à l'autre, on utilise les commandes

SECLECT PRIMARY

SELECT SECONDARY

Cette possibilité se révèle très utile :

— lorsque vous manipulez deux fichiers simultanément (l'emploi d'une seule zone de travail consacrée alternativement à chacun des fichiers conduirait à une gestion plus lourde)

— lorsque vous gérez, par programme, une ''relation'' entre deux fichiers

— En outre, elle est indispensable à l'emploi de la commande JOIN.

9 - La commande JOIN

Elle permet de réaliser ce que l'on nomme une ''jointure'' de deux fichiers. Il s'agit de la création d'un nouveau fichier obtenu en juxtaposant certains champs du premier avec certains champs du second. La juxtaposition a lieu (et donc un nouvel enregistrement est créé) à chaque fois qu'une ''relation'' existe entre un quelconque enregistrement du premier fichier et un quelconque enregistrement du second fichier (Cette relation s'exprime sous forme d'une condition.

CORRECTION DES ''ENTRAÎNEZ-VOUS''
(repérés par un numéro)

CHAPITRE VII

1) • USE STOCK
   • LIST FOR PRIX < 200

2) • USE STOCK
   • LIST PRODUIT, QTE FOR QTE <5

3) • USE REPERT
   • GOTO 1 (superflu si on vient d'ouvrir le fichier)
   • LIST FOR !(NOM) < ''MITENNE''

4) • USE REPERT
   • LIST FOR $(PRENOM,1,1) = ''A''

   où, si on est pas sûr que la première lettre soit écrite en majuscules :

   • LIST FOR ! ($(PRENOM,1,1)) = ''A''

5) • USE REPERT
   • LIST !(NOM) FOR TAILLE > 1.68

6) • USE REPERT

• LIST $(NOM,1,4), $(PRENOM,1,3) FOR $(VILLE,1,1) = ''P''
(ou : $(NOM,1,4) + $(PRENOM,1,3))

7) • USE REPERT

• LIST FOR ''A'' $ NOM

8) • USE REPERT

• LIST FOR ''86'' $ TELE

9) • USE REPERT

• LIST FOR VILLE = ''PARIS'' .AND. $(NOM,1,1) = ''T''

• LIST FOR NOM = ''Albert'' .OR. NOM = ''Claude''

10) • USE STOCK

• LIST FOR CAT = ''A'' .AND. PRIX < 150

• LIST FOR CAT = ''E'' .AND. QTE * PRIX > 14000

CHAPITRE VIII

1) • USE REPERT

• DELETE FOR CODEP < ''46000''

2) • USE REPERT

• DELETE FOR ''L'' $!(NOM)

ou :

• DELETE FOR ''L'' $ NOM .OR. ''I'' $ NOM

3) • GOTO 4

• REPLACE PRENOM WITH ''Michel''

4 • GOTO 1

• REPLACE CODEP WITH ''58100''

5) • GOTO 1

• REPLACE TAILLE WITH 1.76

6) • GOTO 2

• REPLACE PRENOM WITH !(PRENOM)

7) • REPLACE ALLL PRIX WITH PRIX * 1.08

ou :

• REPLACE ALL PRIX WITH PRIX + PRIX * 0.08

8) • REPLACE ALL PRIX WITH PRIX/1.08

9) • REPLACE PRIX WITH PRIX * 0.8 FOR QTE < 20

10) • REPLACE QTE WITH QTE – 5 FOR CAT = ''A''

11) • REPLACE PRIX WITH PRIX * 1.12 FOR QTE > 15 .AND. QTE < 20

CHAPITRE XIV

1)
```
SET TALK OFF
USE REPERT
SUM TAILLE FOR VILLE = ''PARIS'' TO SOMT
COUNT FOR VILLE = ''PARIS'' TO N
STORE SOMT/N TO TMOY
? ''La taille moyenne des parisiens est'', TMOY
SET TALK ON
```

2)
```
SET TALK OFF
USE REPERT
ACCEPT ''Ville choisie'' TO VILLEC
COUNT FOR VILLE = VILLEC TO NH
? ''il y a'', NH, ''personnes habitant'', VILLEC
SET TALK ON
```

3)
```
SET TALK OFF
USE REPERT
ACCEPT ''prénom choisi'' to PR
COUTN FOR PRENOM = PR TO N
SUM TAILLE FOR PRENOM = PR TO SOMT
? ''La taille moyenne des'', PR, ''est'', SOMT/N
```

(Ce programme fonctionne mal quand N contient 0, c'est-à-dire quand aucune personne du fichier ne porte le prénom choisi

```
4)  SET TALK OFF
    USE REPERT
    ACCEPT ''Ville à supprimer'' to VIL
    DELETE FOR VILLE = VIL
    SET TALK ON

5)  SET TALK OFF
    USE STOCK
    ACCEPT ''catégorie choisie'' TO CA
    INPUT ''taux d'augmentation'' TO TAUX
    REPLACE PRIX WITH PRIX * (1 + TAUX) FOR CAT = CA
    SET TALK ON
```

CHAPITRE XV

```
1) SET TALK OFF
   USE STOCK
   ACCEPT ''référence cherchée'' to RF
   LOCATE FOR REF = RF
   IF EOF
      ? ''ce produit ne figure pas au fichier''
    ELSE
      ? ''catégorie'', CAT
      ? ''quantité'', QTE
      ? ''valeur'', PRIX * QTE
   ENDIF
   SET TALK ON
```

CHAPITRE XVI

```
1) SET TALK OFF
   STORE 0 TO NOMBRE
   DO WHILE NOMBRE < > 25
       INPUT ''Choisissez un nombre'' TO NOMBRE
   ENDDO
   ? ''Bravo - c'est ça''
   SET TALK ON
```

Index

à SAY 167
à SAY... GET 170
ACCEPT 138, 140
Ajouter des enregistrements 33
AND (opérateur) 60
APPEND 33
APPEND BLANK 173
APPEND FROM 117

Base de données 3
BOTTOM 87
BROWSE 177

Champ 3
CHANGE 177
CHANGE... FIELDS 177
CHR (fonction) 78
Commentaires 156
Concaténation 49
Conditions 53
Conditions multiples 60
Consulter un fichier 41
CONTINUE 59
COPY STRUCTURE TO 116
COPY TO 65
COUNT 62
CREATE 12
Création 12

DATE (fonction) 178
DBF (type) 14
DELETE 36
DELETE FILE 17
DELETED 39
Démarrage (de dBASE II) 8
DESCENDING 80
DISPLAY 35
DISPLAY MEMORY 134
DISPLAY STATUS 58, 101
DO 128
DO CASE 150
DO WHILE 158

EDIT 31
Effacer 35, 67
Egalité partielle 57
ELSE 143
ENDCASE 150
ENDDO 158
ENDIF 143
EOF (fonction) 161
ERASE 154

Expressions 45
Fiche 3
Fichier 3
FILE (fonction) 178
FIND 84, 88
FOR 53
FORMAT (fichier) 106

GET 170
GOTO 35
GOTO BOTTOM 87
GOTO TOP 87

IF 143
INDEX 83
Index 82
Indexation 82
INPUT 138, 141
INSERT 40
INSERT BEFORE 40
INT (fonction) 178

JOIN 179

LEN (fonction) 178
LIST 21
LIST FILES 14
LIST FILES LIKE 100
LIST OFF 22
LIST STRUCTURE 23
LOCATE FOR 59

Macros 177
Mise à jour 30
Mise à jour 65
Mise à jour (d'un index) 97
Modification de structure 115
MODIFY COMMAND 127
MODIFY STRUCTURE 120

NOT (opérateur) 61

OR (opérateur) 61
Ordre des caractères 77
OTHERWISE 152

PACK 37
PICTURE 178

RANK (fonction) 77
Rapports (édition de) 104
READ 170
RECALL 38
RELEASE 135
RENAME 123
Réorganisation (d'un index) 95
REPLACE 69
REPORT 105
REPORT FORM 107
RESTORE FROM 177
Rubrique 3

SAVE TO 177
SAY 167
SET DEFAULT 15
SET ECHO ON 163
SET EXACT 57
SET INDEX TO 85
SET STEP ON 164
SET TALK OFF 137
SET TALK ON 137
SET TALK ON 163
SORT 76
Sous-totaux 111
STORE 130
STR (fonction) 178
Structure 12
Structure de boucle 158
Structure de choix 143
Structure de choix multiple 150
SUM 63

TOP 87
Tri 75
TRIM (fonction) 50
Type (d'un champ) 13
Type (d'une variable) 133
TYPE (fonction) 178

USE 22
USE... INDEX 85
USING 178

VAL (fonction) 178
Variable 130

Zone de travail 24
Zone primaire 178
Zone secondaire 178
* (fonction) 178
? (commande) 52
! (fonction) 49
$ (condition d'appartenance) 59
$ (fonction) 48

Imprimé en France. — JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° 16404. Dépôt légal : Octobre 1986
N° d'éditeur : 4553

Vous souhaitez utiliser le célèbre logiciel *dBASEII* sur votre *AMSTRAD* (PCW 8256, PCW 8512 ou CPC 6128) !

Ce livre a été réalisé pour vous. Il vous donne les moyens de *piloter* ce logiciel à votre guise pour automatiser la gestion de vos fichiers.

Il vous offre un *apprentissage* très *progressif,* basé sur une *pédagogie active.*

D'une part, vous êtes invité à *expérimenter* chaque nouvelle notion dès qu'elle vous a été présentée.

D'autre part, les nombreuses *manipulations* qui vous sont proposées vous amènent rapidement à la maîtrise de l'outil dBASEII.

En outre, vous êtes systématiquement familiarisé avec les *situations d'erreur* que vous risquez de rencontrer, ce qui vous permettra de mieux vous en protéger.

Ceux d'entre vous qui souhaitent obtenir le maximum de ce progiciel trouveront ici une véritable initiation à la *programmation* en ''langage dBASE''. Elle leur permettra d'aborder la réalisation d'applications ''clés-en-main'' livrables à n'importe quel utilisateur néophyte.

EYROLLES

8338

EYROLLES

dBASE II SUR AMSTRAD

Claude Delannoy

Document numérisé
avec amour par :
AMSTRAD
CPC
MÉMOIRE ÉCRITE
ACME
https://acpc.me/